

**UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA**



TESIS DE GRADO

"PLATAFORMA VIRTUAL BASADA EN AGENTES

PARA COMICIOS ELECTORALES"

**PARA OPTAR AL TÍTULO DE LICENCIATURA EN INFORMÁTICA
MENCIÓN: INGENIERÍA DE SISTEMAS INFORMÁTICOS**

POSTULANTE: RUDY IVÁN BERNAL ESTRADA

TUTOR: M. SC. FÁTIMA CONSUELO DOLZ DE MORENO

REVISOR: Lic. JAVIER REYES PACHECO

LA PAZ- BOLIVIA

2011

RESUMEN

Por lo rustico que resulta el proceso de votación en la actualidad, se vio por conveniente brindar una solución a los problemas que posee, por esta razón se quiso brindar una alternativa con un sistema de votación electrónica, la cual este basada en una nueva alternativa en la programación como lo es la de los agentes móviles, que brindan muchas ventajas que resultan beneficiosas en nuestro país, para reducir el costo de las elecciones eliminando las papeletas impresas en papel y agilizar el proceso de obtención de resultados, resultando implícitamente una colaboración al medio ambiente, también los agentes móviles resultan ventajosos por la utilización de la red de comunicación solo en el momento del envío del agente y no estar conectados todo el tiempo como sucede en un sistema Cliente-Servidor.

El proceso de desarrollo del prototipo se baso en la metodología XP (Extreme Programming), utilizando para la realización de la documentación los diagramas de UML (Unified Modeling Language).

Además se realizó la programación del prototipo de la plataforma con la tecnología de Aglets (Aglets 2.0.2), una tecnología basada en Java perteneciente a IBM pero de distribución libre que brinda la posibilidad de la programación de agentes móviles así como de su monitoreo gracias a su servidor Tahití.

INTRODUCCIÓN

El desarrollo de este trabajo se estructura a partir de la hipótesis, que a lo largo del trabajo se tratara de demostrar, también se da a conocer la problemática y los objetivos del trabajo tratando de justificar el desarrollo de la plataforma de votación por los aspectos principales de costo y tiempo de respuesta.

El capítulo II, hace referencia al marco teórico, el cual es base principal para sustentar la investigación, para ello se seleccionaron los aspectos más importantes de la metodología y herramientas que se utilizaran. Los aspectos más relevantes de este acápite son los agentes móviles, la metodología de desarrollo XP con el lenguaje UML, herramientas de desarrollo de software, etc.

En el siguiente capítulo, se da énfasis a todo lo que significa el proceso de desarrollo del prototipo; desde los requerimientos del usuario hasta finalización de este. Cada una de las actividades que son necesarias para la transformación de los requisitos del usuario en una plataforma, son detalladas de acuerdo a la metodología utilizada (XP) y a otros diagramas especiales desarrollados propiamente para agentes.

Antes de finalizar el trabajo se comprueba la hipótesis tomando en cuenta las principales características denotadas en esta.

Finalmente, se presentan las conclusiones del proyecto y las recomendaciones; incluyendo además la bibliografía y los diferentes anexos.

Índice

1	ANTECEDENTES Y MARCO REFERENCIAL.....	6
1.2	INTRODUCCIÓN	6
1.3	ANTECEDENTES	8
1.3.1	Antecedentes de la votación electrónica	8
1.3.2	Antecedentes del tema de investigación	9
1.4	PLANTEAMIENTO DEL PROBLEMA	11
1.5	OBJETIVOS.....	13
1.5.1	Objetivo General.....	13
1.5.2	Objetivos específicos	13
1.6	LIMITES Y ALCANCES.....	13
1.7	JUSTIFICACIÓN	14
1.8	HIPÓTESIS.....	14
1.9	DISEÑO METODOLÓGICO	14
1.9.1	Metodología de desarrollo	14
1.10	CONSIDERACIONES.....	15
2	MARCO TEÓRICO.....	16
2.2	INTRODUCCIÓN	16
2.3	DEFINICIONES DE AGENTES	16
2.3.1	Componentes de un agente	18
2.3.2	Tipos de agentes	19
2.3.3	Agentes Móviles	21
2.4	METODOLOGÍA XP.....	28
2.4.1	Roles de los actores	29
2.4.2	Fases del Proceso XP	30
2.4.3	Practicas XP	32
2.5	LENGUAJE UNIFICADO DE MODELADO (UML).....	34
2.5.1	Vistas De UML.....	36
2.5.2	Diagramas De UML	37
2.6	AGLETS DE IBM.....	45
3	MARCO APLICATIVO.....	51
3.2	INTRODUCCIÓN	51

3.3	ANÁLISIS DE REQUERIMIENTOS	51
3.3.1	Visión del Sistema	52
3.3.2	Requerimientos Funcionales	54
3.3.3	Requerimientos No Funcionales.....	56
3.4	MODELO DE AGENTES	58
3.4.1	Metodología AOPOA.....	59
3.5	ESPECIFICACIÓN DE LOS AGENTES	60
3.5.1	Agente Servidor de Centralización (MasterC) (Primera Iteración)	60
3.5.2	Agente Centralizador (SlaveC) (Primera Iteración).....	62
3.5.3	Agente Servidor de Votación (MasterServer) (Segunda Iteración)	63
3.5.4	Agente de Usuario (SlaveUser) (Segunda Iteración).....	65
3.6	DISEÑO DE LA PLATAFORMA	67
3.6.1	Diagrama de Capas	67
3.6.2	Especificación de los casos de uso	68
3.6.3	Especificación del Sistema.....	77
3.6.4	Diagramas de Secuencia.....	78
3.6.5	Desarrollo de Interfaces Graficas.....	81
3.7	REQUERIMIENTOS DE HARDWARE Y SOFTWARE.....	85
4	DEMOSTRACIÓN DE LA HIPÓTESIS.....	87
4.2	ANTECEDENTES	87
4.3	COMPROBACIÓN	89
5	CONCLUSIONES Y RECOMENDACIONES	96
	BIBLIOGRAFÍA	99
	ANEXOS	999
	DOCUMENTOS	108

1 ANTECEDENTES Y MARCO REFERENCIAL

1.2 INTRODUCCIÓN

Desde la antigüedad de la sociedad en sí misma, tuvo que considerar las discrepancias del ser humano, cada modo de pensar es diferente en cada persona , lo que deriva en la diversidad de opiniones entre las mismas , pero desde el nacimiento de la democracia, se debió pensar nuevas estrategias y así nacieron las consultas populares, que eran la forma más práctica de mantener el espíritu democrático y entre más era la cantidad de personas se tuvieron que diseñar nuevos métodos para que las opiniones y decisiones de las personas sean escuchadas.

Bolivia en su contexto de refundación después de la Asamblea Constituyente que derivó en la nueva Constitución Política del Estado, en la cual se deja en claro que el pueblo puede tener voz a través de referéndums, elecciones y otros, estipulados en la Nueva Constitución, así también en estos momentos la reestructuración de los poderes del estado

dan como único contexto democrático la realización de este tipo de consultas ciudadanas.

No solo Bolivia como país necesita recurrir a actos democráticos sino toda institución que necesite la participación de gran cantidad de personas para la toma de decisiones vitales para el futuro de la institución misma.

Democracia es una palabra, un concepto, amplio y magnífico. Un concepto muy asociado a los procesos de consulta que resuelven encrucijadas, dirimen expectativas, aprueban propuestas, generan mayorías y minorías, consolidan posiciones, arrojan decisiones y protegen, en definitiva la libertad. Los procesos de consulta son hacedores de comunidad y de responsabilidad social, apuntalan decisiones. No obstante, la laboriosidad y previsión que necesitan dichos procesos, la energía que consume su organización y puesta a punto con las cautelas y garantías que se espera de ellos, diluyen su eficacia práctica.

Los procesos electorales tienen que garantizar que las libertades y derechos de todas las partes están protegidos. Tienen que resolver:

- *La organización del censo*
- *Que el que tiene derecho a votar puede ejercer dicho derecho*
- *Que solo lo hace una vez*
- *Que su voto ha sido válidamente emitido*
- *Que el sistema ha sido auditado por todas las partes*
- *Que los interventores pueden desempeñar eficazmente su papel*
- *Que la anonimidad está protegida*
- *Que la proclamación de resultados es correcta*
- *Que los datos se almacenan correctamente*
- *Que las libertades y derechos de todas las partes están protegidos*

En los procesos electorales cabe distinguir dos niveles. Las consultas o procesos electorales públicos y las consultas o procesos electorales internos de entidades públicas y privadas. La complejidad de uno y otro caso varían notablemente.

1.3 ANTECEDENTES

1.3.1 Antecedentes de la votación electrónica

Al decir *Sufragio Universal* estamos indicando que todos aquellos implicados en la toma de una decisión tienen voz y parte en la toma de la decisión. Universal se toma en el sentido sociológico de la palabra: aquellas personas que se ven implicadas. Esto ilustra una primera división en la tipología de las votaciones:

Votación pública

Es aquella en la que todo el mundo puede -si quiere- participar.

Votación privada

Aquella en la que el universo de votantes está definido de antemano, el sistema de voto electrónico, tiene como objetivo facilitar al elector el ejercicio del voto, eliminando las barreras iniciales que puedan tener algunos votantes ante las nuevas tecnologías.

Los sistemas electorales a nivel mundial constituyen el principal y más importante mecanismo de participación ciudadana e influyen no solo en la participación política o gubernamental sino también de forma relevante en procesos empresariales como la votación en asambleas generales para elección de juntas directivas, entre otras decisiones. Estos sistemas pueden considerarse como sistemas de información susceptibles a la incorporación de Tecnologías de la Información (TIC), que mejoren características como seguridad, verificabilidad y autenticación.

Aunque el voto en papel es el sistema tradicional por excelencia en los países democráticos, las elecciones han ido evolucionando hasta convertirse en sistemas complejos con mayores necesidades de eficiencia y seguridad. En consecuencia, han surgido nuevos mecanismos de votación, algunos mecánicos o que aun necesitan del diligenciamiento manual, y otros electrónicos que incorporan TIC's directamente en el sufragio, dando así lugar al concepto de Votación Electrónica.

La incorporación de la Votación Electrónica presenta entre otras ventajas:

- (i) *El aumento de la eficiencia de las tareas electorales y la precisión en los resultados,*
- (ii) *La automatización de la legitimación de identidad de los votantes,*
- (iii) *Computación de tareas tales como el registro y conteo de votos, el conteo de tarjetones, y la difusión de resultados,*
- (iv) *Mejora en la capacidad para identificar y prevenir situaciones de fraude electoral,*
- (v) *Disminución de la abstención,*
- (vi) *Disminución de votos nulos y de errores en el proceso de votación,*
- (vii) *Aumento en la confiabilidad del proceso de votación, y*
- (viii) *inmediatez del escrutinio.*

1.3.2 Antecedentes del tema de investigación

AGENTES MÓVILES

La programación orientada a objetos representa un mayor nivel de abstracción que la programación basada en procedimientos; es más fácil de comprender y de mantener, por lo tanto, más productiva. La teoría de

agentes establece una serie de mecanismos que pretenden dar un paso más allá en el tratamiento informático distribuido, añadiendo características como la localización o la situación, y permitiendo la interacción dinámica de componentes autónomos y heterogéneos.

Los agentes móviles añaden una singularidad especial al concepto de agentes: la posibilidad de trasladarse de una máquina a otra. Esta característica ofrece ciertas ventajas respecto al tratamiento de la información en modo cliente/servidor, sobre todo en la computación a través de Internet. El auge mundial de “la red de redes” ha permitido el rápido desarrollo de nuevas técnicas inteligentes para búsqueda, filtrado y gestión de datos.

Para comenzar con los agentes móviles, veremos las diferencias principales entre los conceptos de agentes estáticos y agentes móviles, definidos por el O.M.G. en sus Utilidades para la Interoperación entre Sistemas de Agentes Móviles (MASIF).

- **Agente estático:** aquél que sólo puede ejecutarse en la máquina donde fue iniciado. Si éste necesita interactuar con otros agentes o programas o requiere cierta información que no se encuentra en el sistema, la comunicación puede llevarse a cabo mediante cualquier método de interacción para objetos distribuido, como CORBA o RMI de Java.
- **Agente móvil:** aquél que no está limitado al sistema donde se inició su ejecución, siendo capaz de transportarse de una máquina a otra a través de la red. Esta posibilidad le permite interactuar con el objeto deseado de forma directa sobre el sistema de agentes donde se halla dicho objeto. También puede utilizar los servicios ofrecidos por el sistema multiagente destinatario. Ningún sistema de objetos distribuido existente en la actualidad necesita utilizar agentes móviles para comunicarse.

Las tareas de búsqueda y tratamiento de la información en Internet, tienen últimamente una gran importancia en el desarrollo de sistemas basados en agentes móviles. Debido al rápido crecimiento de la Red, el proceso de encontrar los datos más convenientes para un usuario resulta excesivamente tedioso y complejo. En nuestro caso, puede enviarse un agente a los destinos más interesantes para el usuario, localizar y filtrar la información deseada siguiendo las normas dictadas por éste y traerla consigo al ordenador de origen, permitiendo ahorrar tiempo de conexión y ancho de banda, por lo tanto, dinero. Los agentes móviles suelen programarse normalmente en lenguajes interpretados o generadores de código intermedio Telescript, Java, Tcl, ya que éstos dan un mejor soporte a entornos heterogéneos, permitiendo que los programas y sus datos sean independientes de la plataforma utilizada. La **seriación** es el proceso típico por el que se representa el estado completo de un agente mediante una serie que puede ser fácilmente transportada por la red. El proceso de descodificación de dicha serie en el agente se denomina **diseriación**.

1.4 PLANTEAMIENTO DEL PROBLEMA

Durante los últimos años los diferentes actos democráticos se realizan de forma anticuada la cual deja mucho que desear en relación a la confianza, seguridad y rentabilidad de estos métodos, ya que están basados en la utilización de una papeleta impresa en papel, un material rústico, caro y fácil de ser falsificado o adulterado además de necesitar de un conteo uno por uno ,resultando el tiempo necesario para realizar el conteo total de los votos demasiado extenso en relación al tamaño y dispersión de la población electoral, por lo tanto se da la necesidad de la creación de un sistema informático para superar las debilidades que se tienen en la actualidad.

Causas.- las causas del problema son las siguientes

- Los costos de la realización de elecciones, consultas y otros actos democráticos tienen un elevado costo en relación a la producción de las papeletas impresas.
- El sistema que utiliza papeletas impresas es vulnerable a la falsificación y por lo tanto a un posible fraude.
- El tiempo que requiere el conteo de votos es demasiado extenso, haciendo que el resultado total sea dado después de un largo tiempo.
- La confianza del elector es mínima debido a la susceptibilidad que se tiene de posibles fraudes y el no respeto a su voto.

Efectos.- Los efectos que se producen debido al problema son:

- Los resultados finales de las elecciones tienen que esperar demasiado tiempo en ser revelados a la población electora.
- El estado se niega a realizar consultas y elecciones debido a su elevado costo.
- La gente desconfía en los métodos electorales y en las propias elecciones en sí.

Lo anteriormente expuesto nos permite plantear la siguiente interrogante:

¿Podrá una plataforma optimizar el proceso de votación, principalmente en los ámbitos referidos a la seguridad de los datos del sufragio, reducción de costos a largo plazo y brindando rapidez en la unificación de los resultados provistos?

1.5 OBJETIVOS

1.5.1 Objetivo General

Se optimizará el proceso de sufragio y obtención de resultados en actos electorales, desarrollando una plataforma de votación electrónica basada en agentes móviles.

1.5.2 Objetivos específicos

- *Atenuar los costos en material utilizado en las diferentes elecciones, al ser esta plataforma virtual reutilizable y adaptable gracias a la utilización de papeletas electrónicas presentadas en computadoras comunes.*
- *Brindar una solución a la rapidez de la presentación de los resultados después de realizado el acto electoral.*
- *Solucionar los problemas de susceptibilidades de fraude en los comicios electorales brindando resultados parciales en tiempos predeterminados a través de la plataforma virtual.*

1.6 LIMITES Y ALCANCES

El desarrollo de la tesis se limitara al diseño de un prototipo de ejemplificación que pueda probar las características necesarias detalladas en los objetivos planteados como ser la optimización en el manejo del voto electrónico en la red de computadoras provistas para este motivo, demostrar la reducción de los costos con relación a la viabilidad del sistema y de su rentabilidad.

Referente a la programación, diseño y estructuración del sistema se tomara en cuenta principalmente las características de la nueva lógica de programación como lo es la programación orientada a agentes, así, el

prototipo toma las características propias de esta nueva tendencia, para generar ventajas que ofrezcan una alternativa novedosa.

1.7 JUSTIFICACIÓN

El tema a desarrollar en la presente tesis trata de optimizar y solucionar problemas sobre la problemática que frecuentemente tiene que ser tratada como lo es de las justas electorales o consultas a la población por la vía democrática del voto, que en los últimos años se vio disparada por las diferentes consultas que se van realizando, el problema en si es que la movilización de personal y material para realizar estos actos son desmesurados y poco optimizados, que de alguna manera están quedando obsoletos. Es por estos motivos se presenta una propuesta para la realización de una plataforma que pueda servir para realizar los actos electorales por medio del voto electrónico, el cual permitirá ser una respuesta a la optimización del proceso de votación.

1.8 HIPÓTESIS

Es posible optimizar los procesos electorales a través del desarrollo de una plataforma virtual basada en agentes, con lo cual se conseguirá la reducción de costos, reducción en la respuesta del sistema para la obtención de resultados finales de estos procesos.

1.9 DISEÑO METODOLÓGICO

1.9.1 Metodología de desarrollo

Después de haber analizado las diferentes posibilidades y sus respectivas características se tomo la decisión de utilizar una metodología ágil para el

proceso de desarrollo del software específicamente la metodología XP (Extreme Programming) y seguir los siguientes pasos que rigen esta metodología.

1.10 CONSIDERACIONES

Ya planteada la hipótesis se considerará la información relacionada con el tema para la demostración de la misma, como ser:

La Ley Del Órgano Electoral Plurinacional que en su capítulo II de Obligaciones y Atribuciones, en su artículo 23(Obligaciones) en el punto nueve dice textualmente:

“Hacer conocer a la Asamblea Legislativa Plurinacional, en un plazo no mayor a los 30 días, los resultados oficiales de cada proceso electoral, referendo o revocatoria de mandato que haya sido organizado, dirigido, supervisado, administrado o ejecutado por el Órgano Electoral Plurinacional;”.

Esto da cuenta que el proceso de recuento de votos y actas dura más o menos el tiempo estipulado en este artículo, si se considera que en las elecciones pasadas realizadas por la anterior Corte Nacional Electoral el tiempo de entrega de resultados era la misma.

2 MARCO TEÓRICO

2.2 INTRODUCCIÓN

Podemos ver en la actualidad como la tecnología ha avanzado así como la teoría que rige esta tecnología , así también la programación ha evolucionado desde la programación estructurada, la programación orientada a objetos y la programación orientada a eventos, pero en específico una rama en la programación que puede ser importante en el futuro es la programación orientada a agentes, ya que esta permite idealizar nuestro entorno ya no solo como objetos , eventos , sino la unión de estos considerando a cada agente como un programa que puede interactuar con un entorno y reaccionar a ciertos estímulos de este.

2.3 DEFINICIONES DE AGENTES

El termino agente ha sido utilizado en los últimos tiempos para referirse a porciones de software autónomo e inteligente, con ciertos atributos adicionales tales como capacidad de percepción, reacción y deliberación. Sin embargo, no existe aún una definición universalmente

aceptada por los autores de la Inteligencia Artificial del término agente, debido principalmente a la incertidumbre en cuanto a los límites y contextos donde éste puede ser aplicable. Se citan a continuación algunas definiciones del significado del término agente, según varios autores:

- *Un agente es un componente de software y/o hardware que es capaz de actuar de acuerdo a los intereses del usuario [Nwana y Azarmi, 1997].*
- *Un agente es un componente computacional que habita en un ambiente complejo y dinámico. El agente puede percibir y actuar sobre su ambiente. Tiene un conjunto de objetivos o motivaciones que trata de satisfacer por medio de acciones preestablecidas [Maes, 1995].*
- *Un agente es un objeto activo con iniciativa [Parunak, 1996].*
- *Un agente es una entidad que reside en un ambiente donde interpreta datos que reflejan eventos en dicho ambiente y ejecuta comandos que producen efectos en el mismo ambiente. Un agente puede ser puramente software o hardware. En el último caso, una cantidad considerable de software se necesita para que ese hardware sea un agente [The Foundation for Intelligent Physical Agents, 1997].*
- *Un agente es un componente de software capaz de realizar ciertas tareas, que posee determinadas habilidades para la resolución de problemas, posiblemente conteniendo un estado interno y con diferentes propiedades. Un agente no es muy útil como componente aislado, sino que puede formar parte de un sistema multiagente, y desempeñar tareas de acuerdo a sus capacidades de resolución de problemas en colaboración con otros agentes [Moulin y Chaib-draa, 1996].*

- *Un agente es un componente de software activo y persistente que percibe, razona, actúa y se comunica [Huhns y Singh, 1997].*

2.3.1 Componentes de un agente

Una gran cantidad de investigadores de la Inteligencia Artificial asocian a los agentes con conceptos relacionados con los seres humanos, tales como conocimiento, creencia, intenciones y obligaciones [Minsky, 1985; Rao y Georgeff, 1995]. Estos conceptos adicionan algunos atributos a los agentes que se mencionan a continuación:

- *Capacidad de acción: Un agente es capaz de actuar, pudiendo modificar su medio ambiente. La capacidad de acción de un agente está determinada por las acciones que este puede realizar. Por ejemplo, las acciones básicas de un agente encargado de almacenar el contenido de un camión en un depósito podrían ser tomar un elemento del camión, desplazarse y dejar un elemento en el depósito. [Minsky, 1985; Rao y Georgeff, 1995]*
- *Percepción: El agente tiene la habilidad percibir los acontecimientos de su medio ambiente. [Minsky, 1985; Rao y Georgeff, 1995]*
- *Reacción: Los agentes pueden percibir su ambiente y responder de acuerdo a ciertas restricciones temporales a los cambios que ocurren en él. Por ejemplo, un agente podría estar interesado en los cambios de ubicación de otros agentes, y realizar alguna acción en consecuencia. [Minsky, 1985; Rao y Georgeff, 1995]*
- *Autonomía: Es la capacidad de actuar sin intervención humana o de otros sistemas con el fin de alcanzar sus objetivos. Una característica clave de los agentes autónomos es su habilidad de tomar la iniciativa de sus actos, en lugar de actuar sólo en respuesta a su medio ambiente. [Minsky, 1985; Rao y Georgeff, 1995]*

- *Deliberación o racionalidad: Es la capacidad de un agente de actuar determinando qué acciones realizar con el fin de alcanzar sus objetivos. La deliberación a menudo implica que el agente es capaz de aprender, modificando su conocimiento o comportamiento en base a la experiencia, con el objeto de cumplir con sus objetivos de una forma más eficaz y eficiente. [Minsky, 1985; Rao y Georgeff, 1995]*
- *Comunicación a nivel de conocimiento: Representa la capacidad de comunicarse con agentes y personas con lenguajes de alto nivel (análogos a los speech acts humanos, en lugar de protocolos entre programas), tales como KQML (Knowledge Query and Manipulation Language) [Finin et al., 1997].*
- *Habilidad social o cooperación: Los agentes pueden interactuar conjuntamente con otros agentes y/o con humanos para la resolución de un problema. Un ejemplo de esta situación lo constituye un grupo de agentes que simulan un equipo que participa en alguna competencia deportiva [Minsky, 1985; Rao y Georgeff, 1995].*
- *Negociación: Es el proceso de mejora de acuerdos entre dos agentes con respecto a puntos de vista o planes comunes a ambos a través del intercambio de información [Minsky, 1985; Rao y Georgeff, 1995].*
- *Movilidad: Capacidad de moverse autónomamente de una plataforma huésped a otra, posiblemente llevando consigo información acerca del estado de ejecución de las tareas. Pueden llevar directivas o instrucciones que pueden realizarse tanto en forma local como remota [Minsky, 1985; Rao y Georgeff, 1995].*

2.3.2 Tipos de agentes

El conjunto de atributos presentado en la sección anterior ha generado una gran cantidad de clasificaciones de los distintos tipos de agentes, de

acuerdo a diferentes combinaciones de dichos atributos. Una clasificación clásica [Demazeau y Muller, 1991; Nwana, 1996] considera tres tipos de agentes:

- *Reactivos: Los agentes reactivos reaccionan ante los cambios en su ambiente o mensajes de otros agentes. Dichos cambios y mensajes pueden provocar que las acciones o tareas que el agente reactivo posee se activen. Tal activación puede provocar a su vez un cambio en su conocimiento y/o el envío de mensajes a otros agentes. Por otra parte, un agente reactivo no es capaz de razonar acerca de sus intenciones u objetivos [Minsky, 1985; Rao y Georgeff, 1995].*
- *Deliberativos : Son agentes capaces de razonar a partir de sus intenciones (objetivos perseguidos por el agente) y creencias (el conocimiento considerado por el agente como verdadero), y de crear y ejecutar planes de acción a seguir. Pueden seleccionar uno de sus posibles objetivos de acuerdo a sus motivaciones, detectar conflictos y coincidencias en la creación de los planes de acción, etc. [Minsky, 1985; Rao y Georgeff, 1995].*
- *Híbridos: Los agentes híbridos poseen las características de los agentes deliberativos y reactivos en forma simultánea [Minsky, 1985; Rao y Georgeff, 1995].*
- *Una clasificación alternativa muy difundida considera un espacio tridimensional, compuesto por los ejes denominados agencia, inteligencia y movilidad [Bradshaw, 1997]. La agencia es el grado de autonomía del agente, y puede ser medido en función de la naturaleza de sus interacciones con otras entidades que incluyen agentes o incluso humanos. La inteligencia es el grado de razonamiento y aprendizaje que posee un agente. Como mínimo, un agente debe considerar las preferencias del usuario, y almacenar las mismas asociándoles alguna representación. Los mayores niveles de*

inteligencia incluyen un modelo del usuario y razonamiento, con capacidad de aprender y de adaptarse al ambiente. La movilidad se refiere a la capacidad del agente de transportarse entre diferentes sitios de una red [Fuggetta et al., 1998].

2.3.3 Agentes Móviles

En concreto, un agente móvil es un proceso situado en una plataforma o entorno de ejecución para actuar en representación básicamente de una persona u organización, y llevar a cabo una tarea para la cual ha sido especialmente designado. Asimismo, dispone de un conjunto más o menos amplio de atributos entre los que destaca la movilidad en el sentido de que no está limitado al sistema donde comienza la ejecución sino que tiene la completa libertad para viajar, es decir, migrar, interactuar y regresar bajo su propio control.

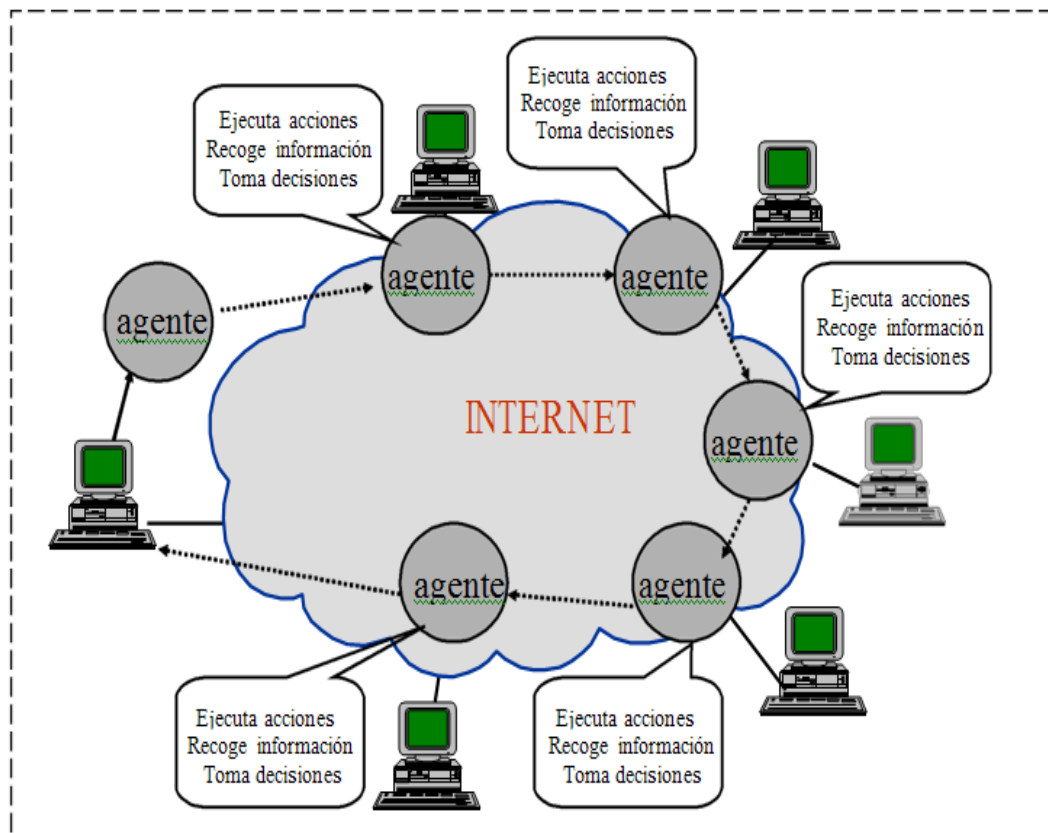


Figura 2.1.-Un agente móvil viajando por la red [Yágüez J, 2010]

Antes de continuar, conviene distinguir entre código móvil y ejecución remota de lo que se entiende por agente móvil y migración. Un código móvil tiene una sola vida, es decir, se transfiere una sola vez antes de su ejecución, la cual se efectúa de forma completa desde su inicio hasta el final. En este contexto, un agente móvil tiene diferentes vidas o “resurrecciones” en el sentido de que puede moverse por un itinerario de entornos de ejecución interactuando con otros agentes o recursos de información y computación hasta llevar a cabo el servicio solicitado; momento en el cual, regresa a la máquina de partida. Como ejemplos de código móvil se pueden citar los siguientes:

- *Applet [Sun Microsystems].- Es una aplicación Java compilada previamente y cuyo código objeto (bytecode) se referencia en una página HTML dentro de un servidor Web. Posteriormente, dicho código objeto se descarga por la red para su interpretación (intérprete Java) y ejecución en un cliente Web.*
- *JavaScript (Netscape).- Es una pequeña aplicación o script hecha en un lenguaje interpretado (lenguaje JavaScript) basado en comandos y cuyo código fuente se referencia en una página HTML dentro de un servidor Web. Seguidamente, dicho código fuente se descarga por la red para su interpretación (intérprete JavaScript) y ejecución en un cliente Web.*
- *Controles ActiveX (Microsoft).- Es una aplicación hecha en un lenguaje cualquiera que se compila previamente y cuyo código objeto se referencia en una página HTML dentro de un servidor Web. Posteriormente, dicho código objeto se descarga por la red para su ejecución directa (sin intérprete) en un cliente Web.*

Los agentes móviles son funcionalmente objetos de software que transportan en un mensaje su estado y código para poder interactuar remotamente con otros agentes o recursos. Para que el agente móvil pueda interactuar con otros agentes o recursos afines, es necesaria la definición de un protocolo que especifique el formato de los mensajes y las acciones que se han de efectuar (Figura 2.2). En el mensaje transportado por el agente móvil se incluye un:

- *Estado:* Como su nombre indica, contiene el estado del objeto o los valores de los atributos del agente y el estado de la máquina o estado de ejecución (contador de programa, punteros de pila, registros,...).
- *Código:* Incluye la clase con los métodos de interacción permitidos.

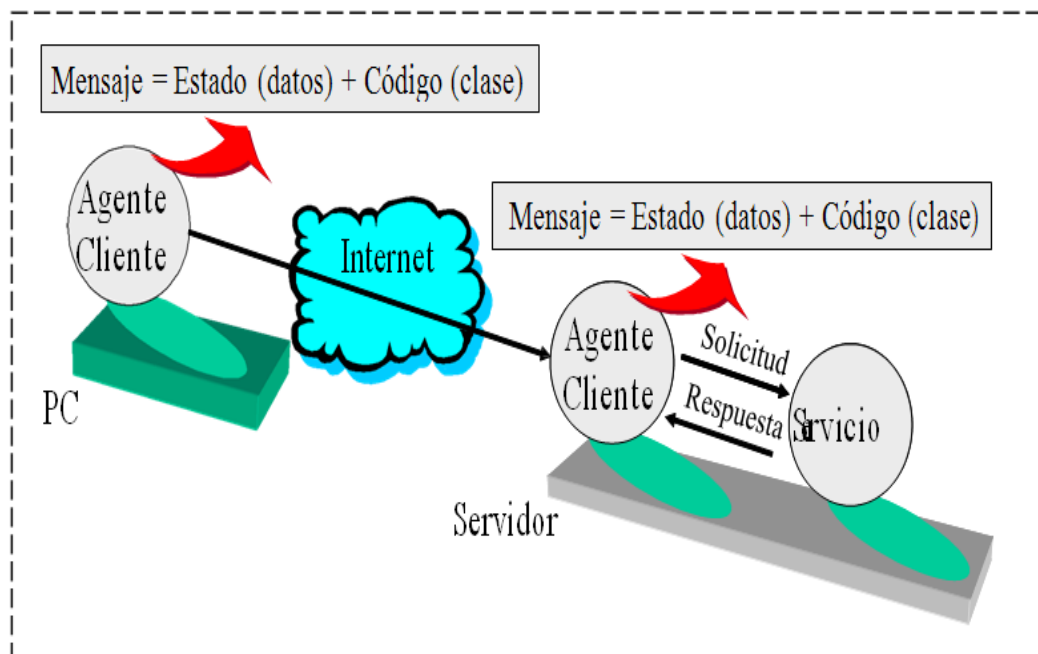


Figura 2.2.- El mensaje transportado por un agente móvil [Yáguez J, 2010]

Como ya se ha indicado, cuando un agente se mueve, realiza una migración suspendiendo la ejecución, transportándose a otra plataforma en

la red y reanudando la ejecución en el punto en que se suspendió. En este contexto, existen dos tipos de migración que se diferencian por el transporte de más o menos información o datos de estado:

- *Migración fuerte: Transporta el estado del objeto (valores de los atributos del agente) y estado de la máquina o estado de ejecución (contador de programa, punteros de pila, registros,...). Es la “foto completa”, justo cuando se suspende la ejecución para su posterior reanudación en el entorno de otra máquina.*

- *Migración débil: Transporta sólo el estado del objeto (valores de los atributos del agente). Es la “foto incompleta” que permite sólo reiniciar la ejecución con los valores actuales de los atributos. Por otro lado, es la típica migración utilizada por los agentes móviles escritos en Java, debido a que este lenguaje no permite la captura del estado de ejecución de un hilo de Java. Se resalta que el modelo de seguridad de la arquitectura JVM (Java Virtual Machina) impide que un programa acceda o manipule directamente sus punteros, registros y pilas. Por consiguiente, cuando un agente Java se transporta de una plataforma a otra, no reanuda su ejecución sino que reinicia ésta con los valores actuales de sus atributos. Evidentemente, para que un agente Java reanudara su ejecución en otra plataforma sería necesario que el programador modificara internamente la máquina JVM realizando las oportunas extensiones o bien simulara el estado de ejecución codificando éste con una serie de variables del programa y un método de arranque.*

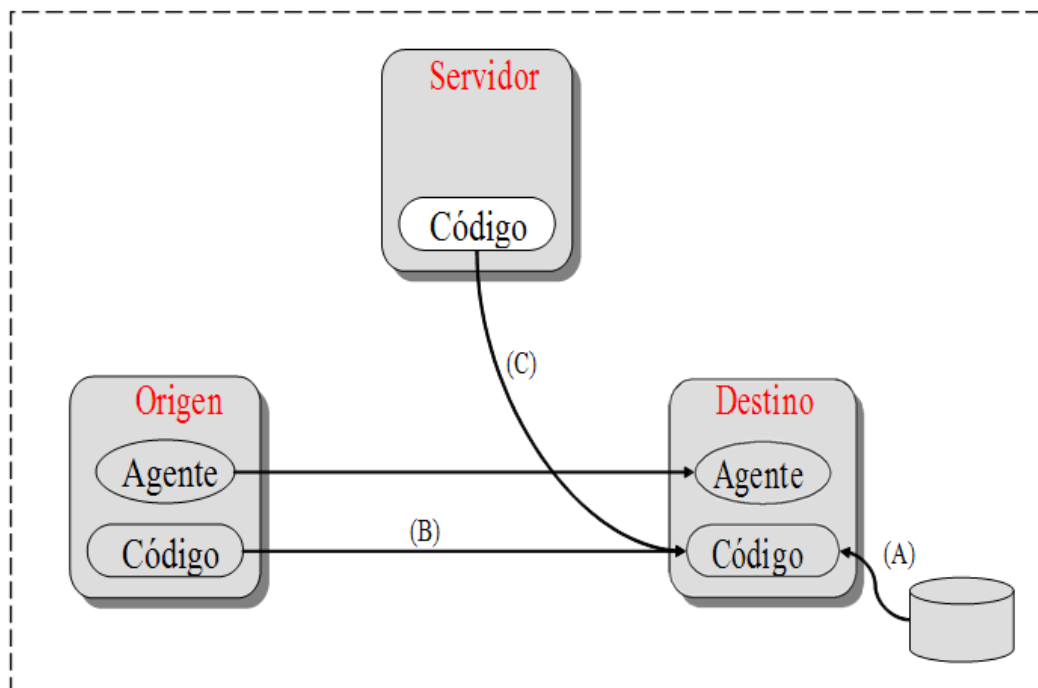


Figura 2.3.- Transferencia del código de la clase de un agente [Yágüez J, 2010]

Con respecto al transporte de la otra parte del mensaje de un agente móvil, es decir, el código de la clase, y según se muestra en la Figura 2.3, se presentan tres posibilidades:

- A. El código se encuentre disponible en la plataforma de destino ya sea en la memoria caché de clases o en el sistema local de ficheros. Consecuentemente, no es necesario transportar dicho código en el mensaje del agente.
- B. El código se encuentre disponible sólo en la plataforma de origen. Este es el caso más frecuente y, por tanto, es necesario su transferencia en el mensaje del agente con la lógica penalización del tráfico de la red.
- C. El código se encuentre en un servidor de clases. Se procede a recuperarlo bajo demanda desde la plataforma de destino.

Llegados a este punto, es importante distinguir ahora entre objetos y agentes móviles. Aunque los agentes son funcionalmente objetos, existen, entre otras, las siguientes diferencias:

- *Un objeto no es un agente móvil ya que un agente móvil es más que un objeto, es decir, es autónomo, flexible, etc.*
- *Un objeto no tiene autonomía sobre su comportamiento a la hora de ejecutar o no un determinado método.*
- *Un objeto no es flexible, es decir, no combina un comportamiento reactivo, proactivo y sociable.*
- *Desde un agente móvil no se invoca un método sino que se realiza una solicitud de servicio que el agente llevará a cabo o no.*

Las distintas acciones que hay que realizar con un agente móvil para que éste pueda migrar desde un entorno de ejecución a otro. Se recuerda que la migración de un agente consiste básicamente en suspender la ejecución, transportarse a otra plataforma en la red y reanudar la ejecución en el punto en que se suspendió. Así en el lado del emisor (plataforma de origen) se suspende la ejecución y se serializa al agente, es decir, se serializa su estado y código en un determinado formato de serialización o de transferencia, es decir, en un flujo de octetos (stream) transferible capaz de ser transportado por la red y restaurado en el lado receptor (plataforma de destino). Seguidamente, se efectúa el marshalling, es decir, se toman los datos serializados y se codifican en un formato o sintaxis común de codificación, representación y transferencia. Finalmente, se transmite al agente, por ejemplo, vía sockets, RPC, Java RMI, CORBA IIOP, etc. A su vez, en el lado del receptor (plataforma de destino) se realiza el proceso contrario o un marshalling, es decir, se decodifican los datos serializados para luego deserializar los octetos de dichos datos al formato nativo de la nueva máquina; con el objetivo final de poder reiniciar (migración débil) o reanudar (migración fuerte) la pertinente ejecución.

Para finalizar con esta introducción y descripción de generalidades sobre agentes móviles, sólo indicar que se pueden aplicar los agentes móviles en muchos escenarios, como por ejemplo:

❖ *Recopilación de información de estado:*

- *El agente móvil recoge información diseminada por la red.*
- *Se establece un itinerario para que un agente móvil viaje secuencialmente por una red de área local recopilando, por ejemplo, información sobre el estado de almacenamiento de los discos duros de las máquinas servidoras de la organización.*

❖ *Búsqueda y Filtrado de Información:*

- *El agente móvil parte de un conocimiento de las preferencias del usuario y elimina la información irrelevante.*

❖ *Monitorización de Noticias:*

- *El agente móvil se envía para que esté a la escucha de ciertos tipos de información y recoja una determinada información diseminada por la red a través del tiempo*

❖ *Copia o mirroring:*

- *El agente móvil realiza una copia inteligente de una información que se actualiza a determinadas horas del día o la noche.*

❖ *Diseminación de la Información:*

- *El agente móvil lleva a cabo una distribución interactiva de noticias o publicidad a grupos interesados.*

❖ *Negociación:*

- *Los agentes móviles negociadores efectúan una planificación de reuniones en nombre de sus representados.*

❖ *Comercio Electrónico:*

- *El agente móvil localiza productos y precios para, finalmente, llevar a cabo una comparación, negociación y compra.*

❖ *Entretenimiento:*

- *El agente móvil se programa con una determinada estrategia para su posterior envío a un servidor de juegos. Opcionalmente, se puede establecer todo un itinerario de servidores de juegos.*
- ❖ *Subastas:*
 - *El agente móvil se programa con una determinada estrategia para su posterior envío a un servidor de subastas. Opcionalmente, se puede establecer todo un itinerario de servidores de subastas.*

2.4 METODOLOGÍA XP

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico [Beck K., 2000].

Los principios y prácticas son de sentido común pero llevadas al extremo, de ahí proviene su nombre. Kent Beck, el padre de XP, describe la filosofía de XP en sin cubrir los detalles técnicos y de implantación de las prácticas. Posteriormente, otras publicaciones de experiencias se han encargado de dicha tarea. A continuación presentaremos las características esenciales de XP organizadas en los tres apartados siguientes: historias de usuario, roles, proceso y prácticas.

2.4.1 Roles de los actores

Aunque en otras fuentes de información aparecen algunas variaciones y extensiones de roles XP, en este apartado describiremos los roles de acuerdo con la propuesta original de Beck [Beck K., 2000].

Programador

El programador escribe las pruebas unitarias y produce el código del sistema. Debe existir una comunicación y coordinación adecuada entre los programadores y otros miembros del equipo [Beck K., 2000].

Cliente

El cliente escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio. El cliente es sólo uno dentro del proyecto pero puede corresponder a un interlocutor que está representando a varias personas que se verán afectadas por el sistema [Beck K., 2000].

Encargado de pruebas (Tester)

El encargado de pruebas ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas [Beck K., 2000].

Encargado de seguimiento (Tracker)

El encargado de seguimiento proporciona realimentación al equipo en el proceso XP. Su responsabilidad es verificar el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, comunicando los resultados para mejorar futuras estimaciones [Beck K., 2000].

También realiza el seguimiento del progreso de cada iteración y evalúa si los objetivos son alcanzables con las restricciones de tiempo y recursos

presentes. Determina cuándo es necesario realizar algún cambio para lograr los objetivos de cada iteración.

Entrenador (Coach)

Es responsable del proceso global. Es necesario que conozca a fondo el proceso XP para proveer guías a los miembros del equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente [Beck K., 2000].

Consultor

Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto. Guía al equipo para resolver un problema específico [Beck K., 2000].

Gestor (Big boss)

Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación [Beck K., 2000].

2.4.2 Fases del Proceso XP

El ciclo de vida ideal de XP consiste de seis fases: Exploración, Planificación de la Entrega (Release), Iteraciones, Producción, Mantenimiento y Muerte del Proyecto [Beck K., 2000].

Fase I: Exploración

En esta fase, se plantea a grandes rasgos las herramientas, tecnologías y prácticas que se utilizarán en el proyecto, y se exploraran las posibilidades de la arquitectura del sistema

Fase II: Planificación de la Entrega

En esta fase de la planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias.

Fase III: Iteraciones

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto.

Fase IV: Producción

La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase.

Fase V: Mantenimiento

Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte.

Fase VI: Muerte del Proyecto

Esta fase solo se hará mención según el desarrollo del proyecto y en caso de que ocurriera algún acontecimiento que imposibilite la finalización del mismo.

2.4.3 Practicas XP

La principal suposición que se realiza en XP es la posibilidad de disminuir la mítica curva exponencial del costo del cambio a lo largo del proyecto, lo suficiente para que el diseño evolutivo funcione. XP apuesta por un crecimiento lento del costo del cambio y con un comportamiento asintótico. Esto se consigue gracias a las tecnologías disponibles para ayudar en el desarrollo de software y a la aplicación disciplinada de las prácticas que describiremos a continuación [Wake W., 2002]

El juego de la planificación

Es un espacio frecuente de comunicación entre el cliente y los programadores. El equipo técnico realiza una estimación del esfuerzo requerido para la implementación de las historias de usuario y los clientes deciden sobre el ámbito y tiempo de las entregas y de cada iteración. Esta práctica se puede ilustrar como un juego, donde existen dos tipos de jugadores: Cliente y Programador. El cliente establece la prioridad de cada historia de usuario, de acuerdo con el valor que aporta para el negocio. Los programadores estiman el esfuerzo asociado a cada historia de usuario. Se ordenan las historias de usuario según prioridad y esfuerzo, y se define el contenido de la entrega y/o iteración, apostando por enfrentar lo de más valor y riesgo cuanto antes. Este juego se realiza durante la planificación de la entrega, en la planificación de cada iteración y cuando sea necesario reconducir el proyecto [Wake W., 2002].

Entregas pequeñas

La idea es producir rápidamente versiones del sistema que sean operativas, aunque obviamente no cuenten con toda la funcionalidad pretendida para el sistema, pero sí que constituyan un resultado de valor para el negocio. Una entrega no debería tardar más 3 meses [Wake W., 2002].

Metáfora

En XP no se enfatiza la definición temprana de una arquitectura estable para el sistema. Dicha arquitectura se asume evolutiva y los posibles inconvenientes que se generarían por no contar con ella explícitamente en el comienzo del proyecto se solventan con la existencia de una metáfora. El sistema es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo. Una metáfora es una historia compartida que describe cómo debería funcionar el sistema. Martin Fowler explica que la práctica de la metáfora consiste en formar un conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema. Este conjunto de nombres ayuda a la nomenclatura de clases y métodos del sistema [Wake W., 2002].

Diseño simple

Se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto. La complejidad innecesaria y el código extra debe ser removido inmediatamente. Kent Beck dice que en cualquier momento el diseño adecuado para el software es aquel que: supera con éxito todas las pruebas, no tiene lógica duplicada, refleja claramente la intención de implementación de los programadores y tiene el menor número posible de clases y métodos [Wake W., 2002].

Pruebas

La producción de código está dirigida por las pruebas unitarias. Las pruebas unitarias son establecidas antes de escribir el código y son ejecutadas constantemente ante cada modificación del sistema. Los clientes escriben las pruebas funcionales para cada historia de usuario que deba validarse. En este contexto de desarrollo evolutivo y de énfasis en pruebas constantes, la automatización para apoyar esta actividad es crucial [Wake W., 2002].

Refactorización (Refactoring)

La refactorización es una actividad constante de reestructuración del código con el objetivo de remover duplicación de código, mejorar su legibilidad, simplificarlo y hacerlo más flexible para facilitar los posteriores cambios. La refactorización mejora la estructura interna del código sin alterar su comportamiento externo. Robert Martin señala que el diseño del sistema de software es una cosa viviente. No se puede imponer todo en un inicio, pero en el transcurso del tiempo este diseño evoluciona conforme cambia la funcionalidad del sistema. Para mantener un diseño apropiado, es necesario realizar actividades de cuidado continuo durante el ciclo de vida del proyecto [Wake W., 2002].

De hecho, este cuidado continuo sobre el diseño es incluso más importante que el diseño inicial. Un concepto pobre al inicio puede ser corregido con esta actividad continua, pero sin ella, un buen diseño inicial se degradará.

Estándares de programación

XP enfatiza la comunicación de los programadores a través del código, con lo cual es indispensable que se sigan ciertos estándares de programación (del equipo, de la organización u otros estándares reconocidos para los lenguajes de programación utilizados). Los estándares de programación mantienen el código legible para los miembros del equipo, facilitando los cambios [Wake W., 2002].

2.5 LENGUAJE UNIFICADO DE MODELADO (UML)

“UML” son las siglas de Unified Modeling Language. Es un lenguaje estándar para el modelado de software – un lenguaje para la visualización, especificación, construcción y documentación de los artefactos de sistemas en los que el software juega un papel importante [Booch y Rumbaugh, 2000]. EL Proceso Unificado utiliza el estándar de modelado visual UML.

Básicamente, UML permite a los desarrolladores visualizar los resultados de su trabajo en esquemas o diagramas estandarizados.

UML (Unified Modeling Language) [Booch y Rumbaugh, 2000], es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema de software orientado a objetos, esta descripción no pretende ser exhaustiva en términos sintácticos, semánticos y de presentación de los elementos de la notación.

Área	Vista	Diagramas	Conceptos principales
Estáticos	Vista estática	Diagrama de clases	Clase, asociación, generalización, dependencia, realización, interfaz.
	Vista de casos de uso	Diagrama de casos de uso	Casos de uso, actor, asociación, extensión, inclusión, generalización de casos de uso.
	Vista de implementación	Diagrama de componentes	Componente, realización, interfaz, dependencia.
	Vista de despliegue	Diagrama de despliegue	Nodo, componente, dependencia, localización.
Dinámicos	Vista de máquinas de estado	Diagrama de estados	Estado, evento, transición, acción.
	Vista de actividades	Diagrama de actividad	Estado, actividad, transición, determinación, división, unión.
	Vista de interacción	Diagrama de colaboración	Colaboración, interacción, rol de colaboración, mensaje.
		Diagrama de secuencia	Interacción objeto, mensaje, activación
Gestión del modelo	Vista de gestión del modelo	Diagrama de clases	Paquete, subsistema, modelo.

Tabla 2.1 Vistas y Diagramas UML

Debe entenderse como una guía inicial al tema. Se agrupan los conceptos básicos por tipo de diagrama.

- 1) Diagrama de estructura estática
- 2) Diagrama de casos de uso
- 3) Diagrama de Secuencia
- 4) Diagrama de Colaboración
- 5) Diagrama de Estados
- 6) Diagrama de Actividades
- 7) Diagrama de Implementación

2.5.1 Vistas De UML

Las vistas que se citan a continuación, están muy relacionadas con la metodología a emplear en el desarrollo mismo del software. Según Jacobson, Booch, y Rumbaugh se tienen las siguientes vistas [Booch y Rumbaugh, 2000]:

- **Vista estática**, La vista estática modela los conceptos del dominio de la aplicación, así como los conceptos internos inventados como parte de la implementación de la aplicación.
- **Vista de los casos de uso**, Modela funcionalidad del sistema según como lo perciben los usuarios externos, llamados *actores*.
- **Vista de implementación**, modela los componentes de un sistema así como las dependencias entre los componentes, para poder determinar el impacto de un cambio propuesto. También modela la asignación de clases y de otros elementos del modelo a los componentes,
- **Vista de despliegue**, representa la disposición de las instancias de componentes de ejecución en instancias de nodos. Un nodo es un recurso de ejecución, tal como una computadora, un dispositivo, o memoria. Esta vista permite determinar las consecuencias de la

distribución y de la asignación de recursos. Tanto la vista de implementación como la de despliegue son conocidas como vistas físicas.

- **Vista de máquina de estados,** Una maquina de estados, modela las posibles historias de vida de un objeto de una clase. Una maquina de estados contiene los estados conectados por transiciones. Cada estado modela un periodo de tiempo durante la vida de un objeto, en el que satisface ciertas condiciones.
- **Vista de actividades,** Un grafo de actividades es una variante de una maquina de estados, que muestra las actividades de computación implicadas en la ejecución de un cálculo. También describe grupos secuenciales y concurrentes de actividades.
- **Vista de interacción.** Describe secuencias de intercambios de mensajes entre los roles que implementan el comportamiento de un sistema. Un rol de clasificador o simplemente rol, es la descripción de un objeto, que desempeña un determinado papel dentro de una interacción, distinto de otros objetos de la misma clase. Esta visión proporciona un vista integral del comportamiento del sistema.

2.5.2 Diagramas De UML

2.5.2.1 Diagramas de Casos de Uso.

Un caso de uso es una descripción de las acciones de un sistema desde el punto de vista del usuario. Para los desarrolladores del sistema, ésta es una herramienta muy valiosa, puesto que es una técnica para especificar la funcionalidad y el comportamiento de un sistema mediante su interacción con los usuarios y/o otros sistemas. Es decir ayuda a obtener los requerimientos desde el punto de vista de los usuarios.

Descripción de los elementos de un modelo de casos de uso:

Actor, es una entidad externa del sistema que de alguna manera participa en la historia del caso de uso.



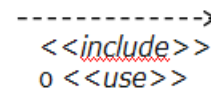
Caso de uso, se representa mediante una elipse.



Asociación, elemento que conecta el actor con el caso de uso.



Inclusión, Si los pasos en una situación dentro de un caso de uso son los mismos que los de otro, entonces podemos adicionar otro caso de uso que incluya los pasos similares.



Extensión, un caso de uso extiende al original dado que agrega otros pasos a la secuencia del caso de uso existente.

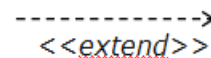


Figura 2.4 Elementos de un caso de uso

2.5.2.2 Diagramas de clases

Un diagrama de clases muestra un conjunto de clases, interfaces y colaboraciones y las relaciones entre éstos; los diagramas de clases muestran el diseño del sistema desde un punto de vista estático

Algunos elementos que intervienen en los diagramas de clases son los siguientes [Schmuller J.]:

Clase, es una categoría o grupo de cosas que tienen atributos y acciones similares.

Atributo, es una propiedad o característica de una clase.

Métodos u operaciones, es algo que la clase puede realizar, o que usted (u otra clase) pueden hacer a una clase

Asociaciones, se conoce como asociación cuando las clases se conectan entre sí de forma conceptual.

Multiplicidad, es la cantidad de objetos de una clase que se relacionan con un objeto de la clase asociada.

Agregación, Es un tipo especial de asociación que especifica una relación todo-parte entre el agregado (el todo) y una parte componente (la parte).

Herencia o generalización, Es una relación entre una superclase que hereda sus características (atributos y operaciones) y subclases que harán suyas dichas características.

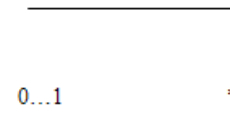
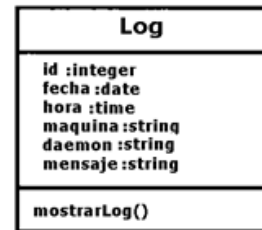


Figura 2.5 Elementos de un Diagrama de Clases [Schmuller J.]

2.5.2.3 Diagrama de objetos

Es un diagrama que muestra un conjunto de objetos y sus relaciones en un momento determinado; los diagramas de objetos muestran el diseño o los procesos de un sistema desde un punto de vista estático[Schmuller J.].

Un objeto es una instancia de una clase (una entidad que tiene valores específicos de los atributos y acciones). En la siguiente figura se muestra la representación de un objeto:

El símbolo es un rectángulo, con el nombre subrayado. La instancia esta a la izquierda de los dos puntos (:), y el nombre de la clase a la derecha.

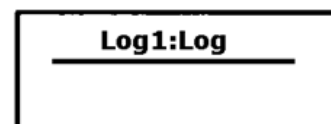


Figura 2.6 Representación de un objeto [Schmuller J.]

2.5.2.4 Diagrama de estados

Representan la secuencia de estados por los que un objeto o una interacción entre objetos pasa durante su tiempo de vida en respuesta a estímulos (eventos) recibidos. Representa lo que en conjunto se puede denominar, una maquina de estados.

Cuando un objeto o una interacción pasa de un estado a otro por la ocurrencia de un evento se dice que ha sufrido una transición; existen varios tipos de transición entre objetos; simples (normales y reflexivas) y complejas. Además una transición puede ser interna si el estado del que parte el objeto o interacción es el mismo que al que llega, no se provoca un cambio de estado y se representan dentro del estado, no de la transición [Schmuller J.].

Simbología

El icono para el estado es un rectángulo de vértices redondeados, y el símbolo de una transición es una línea continua y una punta de flecha. Un círculo relleno se interpreta como el punto inicial de la secuencia y, una diana representa al punto final.

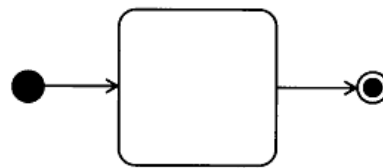


Figura 2.7 Representación de un Diagrama de Estado [Schmuller J.]

2.5.2.5 Diagrama de secuencia

Muestran las interacciones entre un conjunto de objetos, ordenadas según el tiempo en que tienen lugar. En los diagramas de este tipo intervienen objetos, que tienen un significado parecido al que tienen en los diagramas de colaboración, es decir son instancias concretas que participan en la interacción. El objeto puede existir sólo durante la ejecución de la interacción. Un diagrama de secuencia representa una forma de indicar el periodo durante el que un objeto está desarrollando una acción directamente o a través de un procedimiento.

En este tipo de diagramas intervienen los mensajes, que son la forma en que se comunican los objetos: el objeto origen solicita (llama a) una operación del objeto destino. Existen distintos tipos de mensajes: simples, asíncronos y síncronos.

Los diagramas de secuencia permiten indicar cuál es el momento en el que se envía o se completa un mensaje mediante el tiempo de transición, que se especifica en el diagrama [Schmuller J.].

En un diagrama de secuencia los objetos se colocan de izquierda a derecha en la parte superior. Cada línea de vida de un objeto es una línea discontinua que se desplaza hacia abajo del objeto. Una flecha conecta a una línea de vida con otra, y representa un mensaje de un objeto a otro. Junto con la línea de vida se encuentra un pequeño rectángulo conocido como activación, el cual representa la ejecución de una operación que realiza el objeto.

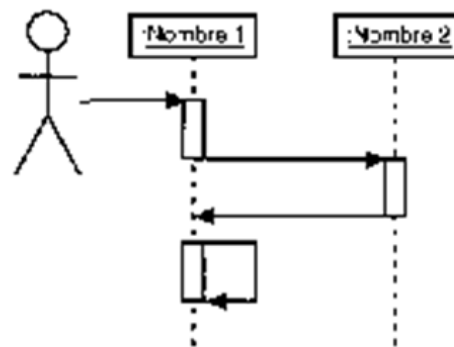


Figura 2.8 Representación de un Diagrama de Secuencias [Schmuller J.]

2.5.2.6 Diagrama de colaboración

Muestra la interacción entre varios objetos y los enlaces que existen entre ellos. Representa las interacciones entre objetos organizados alrededor de los objetos y sus vinculaciones. A diferencia de un diagrama de secuencia, un diagrama de colaboraciones muestra las relaciones entre los objetos, no la secuencia en el tiempo en que se producen los mensajes. Ambos diagramas de colaboración y secuencia expresan información similar, pero en un forma diferente [Schmuller J.].

En un diagrama de colaboración, para representar un mensaje, dibujará una flecha cerca de la línea de asociación que apuntará al objeto receptor. El tipo de mensaje se mostrará cerca de la flecha

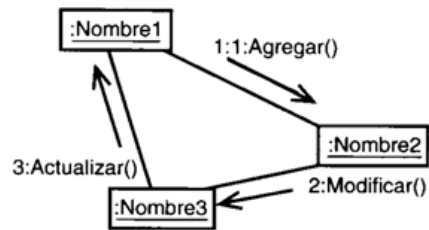


Figura 2.9 Representación de un Diagrama de Colaboración [Schmuller J.]

2.5.2.7 Diagrama de actividades

El diagrama de actividad del UML, es muy parecido a los viejos diagramas de flujo. Le muestra los pasos (conocidos como actividades) así como de decisión y bifurcaciones. Es útil para mostrar lo que ocurre en un proceso de negocios u operación [Schmuller J.].

A cada actividad se representa por un rectángulo angosto con las esquinas redondeadas. Una flecha representa la transición de una a otra actividad. Cuenta con un punto inicial y final.



Figura 2.10 Representación de un Diagrama de Actividades [Schmuller J.]

2.5.2.8 Diagrama de componentes

Muestra un conjunto de componentes y sus relaciones. Un componente de software es una parte física de un sistema, y se encuentra en la computadora, no en la mente del analista. Puede tomarse como un componente: una tabla, archivo de datos, ejecutables, documentos, y etc. [Schmuller J.].

Existen tres tipos de componentes:

Componentes de distribución, conforman el fundamento de los sistemas ejecutables (p.e. DLL, ejecutables, controles ActiveX)

Componentes para trabajar en el producto, a partir de los cuales se han creado los componentes de distribución (archivos de bases de datos y de código).

Componentes de ejecución, creados como resultado de un sistema en ejecución.

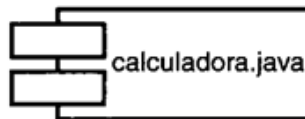


Figura 2.11 Símbolo que representa a un Componente [Schmuller J.]

2.5.2.9 Diagrama de plataformas o despliegue

Ilustra la forma en que luce un sistema físicamente cuando sea conjugado. Muestra la configuración de los componentes de hardware, los procesos, los elementos de procesamiento en tiempo de ejecución y los objetos que existen en tiempo de ejecución. En este tipo de diagramas intervienen nodos, asociaciones de comunicación, componente dentro de los nodos y objetos que se encuentran a su vez dentro de los componentes. Un nodo es un objeto físico en tiempo de ejecución, es decir una maquina que se compone habitualmente de, por lo menos, memoria y capacidad de procesamiento, a su vez puede estar formada por otros componentes [Schmuller J.].

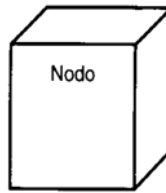


Figura 2.12 Representación de un nodo en un Diagrama de Despliegue
[Schmuller J.]

Herramientas de Desarrollo del Software

La arquitectura del Aglets Software Development Kit (ASDK) presenta una herramienta de desarrollo para agentes móviles, creada por los laboratorios de I.B.M. en Tokio, siendo una de las primeras en participar en las especificaciones MASIF para CORBA. Un aglet es un objeto escrito en Java capaz de visitar máquinas que soporten el entorno de ejecución Aglets WorkBench (AWB). Dicho entorno permite:

- Cifrar el código y los datos de un *aglet* utilizando el método de seriación de Java (JOS).
- Trasladar agentes utilizando el Protocolo para el Transporte de Aglets (ATP).
- Ofrecer un Interfaz de Programación para Aglets (A-API).
- Interconexión e intercambio de información entre *aglets* y otros objetos mediante paso de mensajes.
- El ciclo de vida de un *aglet* puede tratarse por métodos basados en captura de eventos. Los eventos definidos son: creación, clonación, expedición, retractación, eliminación, activación, desactivación y paso de mensaje.
- Control de seguridad mediante definición de autoridades y de sus privilegios y preferencias.

- Los elementos principales de la arquitectura *aglet* son Aglet: define los métodos básicos para el control de un agente móvil.
- Representante: aísla al *aglet* del entorno, permitiendo un mayor grado de seguridad y ofreciendo transparencia respecto a la localización del agente.
- Contexto: ofrece al *aglet* un interfaz con su entorno de operación.
- Mensaje: objetos utilizados para la comunicación entre *aglets*.

2.6 AGLETS DE IBM

IBM de Japón ha desarrollado el concepto de aglet basado en Java. El aglet extiende el modelo de código móvil que presentan los applets de Java. Así como sucede en un applet, los archivos .class de los aglets pueden migrar a través de las redes, pero cuando los Aglets migran también lo hace su estado. Un applet es código que puede moverse a través de las redes desde un servidor hacia el cliente. Un aglet es un programa Java (código y estado) que puede moverse desde una máquina anfitriona hacia otra, pero no solo eso, sino que pueden visitar varias máquinas anfitrionas secuencialmente o al azar y regresar luego a la máquina desde la cual migró inicialmente si así es el diseño [Lange y Oshima, 1998].

Un aglet para su ejecución requiere una aplicación anfitriona Java o servidor de agentes, ejecutándose en cada computadora antes de que el aglet pueda visitar la máquina. Cuando los aglets viajan a través de la red, migran desde un anfitrión de aglets a otro. Cada anfitrión ejecuta un administrador de seguridad para reforzar restricciones sobre las actividades de aglets riesgosos. Los anfitriones cargan aglets a través de cargadores tipo class que saben cómo recuperar los archivos .class así como el estado desde un anfitrión remoto.

El Aglets Workbench (AWB) es el conjunto de clases y herramientas necesarias para la construcción de agentes móviles. El AWB está integrado por [Lange y Oshima, 1998]:

1. Aglet Framework.
2. Agent Transfer Protocol.
3. Tazza.
4. Tahiti.
5. Fiji.

Aglet Framework. El componente principal del AWB, es el Aglet Framework, este está basado en el lenguaje de programación Java. Este ambiente introduce el concepto de aglet (agent applet), un agente móvil escrito en Java que es una clase abstracta de agente móvil.

El Aglet Framework provee un mecanismo de seguridad extensible. El lenguaje Java provee la primera capa de seguridad ya que el verificador bytecode de Java checa el formato de código y genera chequeos de consistencia sobre este código. La siguiente capa de protección es el administrador de seguridad (Security Manager) que permite a los mismos programadores establecer sus propios mecanismos de protección. La última capa de protección es el mismo API de seguridad de Java, esta interfaz de programación permite criptografía, firmas digitales, encriptación y autenticación [Lange y Oshima, 1998].

Agent Transfer Protocol. Para hacer posible la transferencia de agentes, los aglets utilizan un protocolo de transferencia de agentes, el ATP y actualmente está disponible la primera versión de este llamado ATP/O. 1. Este protocolo está basado en los URL's, y se enfoca principalmente a Internet y ofrece servicios independientes de la plataforma y de lenguaje. ATP posee clases altamente portables que proveen un API estándar para

crear servidores ATP, para conectar sitios ATP y generar respuestas a requerimientos ATP.

Tazza. En la actualidad existen varios compiladores que integran todo un ambiente de desarrollo visual como Visual Basic, PowerBuilder y Visual C++. Este ambiente de desarrollo en el AWB se llama Tazza. Este componente aun no se ha liberado en las versiones del AWB y se promete que estará disponible en versiones subsecuentes [Lange y Oshima, 1998].

Tahiti. El AWB contiene un administrador visual de agentes y trabaja conjuntamente con el aglet framework, este componente es el anfitrión del aglet. Este administrador es una interfaz gráfica hecha en Java que permite monitorear y controlar la ejecución de aglets. Es el componente esencial para poder utilizar los aglets, ya que abre un puerto de comunicaciones, como el que utilizan los demonios HTTP que por lo común es el puerto 80. Esto es necesario para que el ATP pueda contactar otros sitios que funcionan como este protocolo [Lange y Oshima, 1998].

Fiji. Fiji es capaz de crear applets (Fiji applets) los cuales corren en contextos de aglets y puede crear, despachar o retractar aglets desde páginas Web. Fiji se compone de una biblioteca de aglets Fiji que es una biblioteca de clases AWB extendidas al visualizador Web, similar a un plugin y de un Kit Fiji que es una biblioteca class y un módulo de ruteo. Ambos deben ser instalados en el servidor de Web.

El J-ATCI IBM introdujo toda una tecnología para hacer posible el envío de agentes móviles a través de las redes de una manera estándar. El J-ATCI es una de las partes fundamentales del AWB.

El J-ATCt (Java Aglet Transfer and Communication Interface) es un estándar independiente del protocolo para comunicar o mover agentes dentro de una red. La implementación de J-ATCI es totalmente ajena a la interfaz, esta consiste de dos paquetes Java: el paquete atci con sus clases

e interfaces y una implementación para una red en especial xxx.atci. IBM tiene el ibm.atci que soporta los protocolos ATP y HTTP, y estas son sus ventajas:

- Simplicidad: El J-ATCI es fácil de usar.
- Extensibilidad: El J-ATCI es abierto a cualquier protocolo de comunicación.
- Independencia del sistema: Los agentes implementados en J-ATCI son capaces de ser transferidos a otros nodos y de comunicarse entre ellos.

Conceptos

Un contexto es el lugar de trabajo de un aglet. Es un objeto estacionario que provee los medios para la administración de los aglets en un ambiente de ejecución uniforme en donde el sistema anfitrión está seguro en contra de agentes maliciosos. Un nodo en una red puede tener múltiples contextos [Lange y Oshima, 1998].

Un proxy es el representante de un aglet. Este sirve como un escudo para el aglet al cual protege del acceso directo hacia sus nodos públicos. El objeto proxy provee transparencia en la localidad del aglet [Lange y Oshima, 1998].

Un mensaje es un objeto que es intercambiado entre aglets. Este puede ser pasado como parámetro de una forma síncrona o asíncrona entre aglets, por tanto puede ser usado por los aglets para la colaboración e intercambio de información [Lange y Oshima, 1998].

Un administrador de mensajes permite el control de concurrencia en la administración de mensajes por parte de un aglet [Lange y Oshima, 1998].

Un itinerario es el plan de viaje de un aglet [Lange y Oshima, 1998].

Un identificador es un objeto atado a cada aglet. Este identificador es globalmente único e inmutable a lo largo de la vida del aglet [Lange y Oshima, 1998].

El J–AAPI. El Java Aglet Application Programming Interface (J–AAPI) se ha propuesto como un estándar para el desarrollo de agentes móviles basados en Java. Este API provee los mecanismos para la creación, inicialización, manejo de mensajes, despacho, recuperación, activación, desactivación, generación de clones, o destrucción de agentes [Lange y Oshima, 1998].

El J–APPI está integrado por lo siguiente:

Interfaces:

- AgletContext. Genera un contexto en el cual el aglet es creado, desactivado, eliminado entre otras acciones. Es usada por un aglet para obtener información acerca de su ambiente y mandar mensajes hacia el ambiente, incluyendo otros agentes que actualmente se encuentran activos en ese ambiente. Provee medios para mantenimiento y administración de los aglets en ejecución. Los contextos son típicamente creados por el sistema, el cual tiene un demonio activado en cierto puerto, esperando la llegada de un nuevo aglet o para asistirlo en su proceso de migración. Los aglets que arriban son insertados en el contexto por medio del demonio [Lange y Oshima, 1998].
- Message Manager
- Future Reply

Clases:

- Aglet. Está es una clase abstracta que se utiliza como clase base cuando se desarrolla un programa aglet. Esta clase define métodos para controlar el ciclo de vida del aglet, como son creación, clonación, despacho, retracción, desactivación, activación, destrucción, mensajes [Lange y Oshima, 1998].

- **AgletIdentifier.** A cada aglet se le es asignado un identificador único el cual es preservado por el aglet durante todo su ciclo de vida. Esta clase es una abstracción conveniente para su identidad. El objeto identificador esconde la representación específica de implementación de la identidad del aglet [Lange y Oshima, 1998].
- **AgletProxy.** Para la interacción entre Aglets, un aglet normalmente no invoca los métodos de otro aglet directamente. Para lograr eso hace uso de un objeto AgletProxy, el cual sirve como representante del aglet. La clase AgletProxy contiene métodos que permiten a los aglets hacer requerimientos sobre otros aglets para que estos realicen una acción. Estas acciones son `dispatch()`, `clone()`, `deactive()`, `dispose()` [Lange y Oshima, 1998].
- **Itinerary.** Al proveer a un aglet con un itinerario se provee una forma apropiada para hacer viajes con múltiples destinos. Esta es una clase abstracta por tanto no puede instanciarse directamente. Así que debe crearse una subclase y heredar en esta la clase Itinerary [Lange y Oshima, 1998].
- **Message.** Una importante propiedad de los aglets es que se pueden comunicar. La comunicación entre agentes es soportada de modo que intercambiar mensajes, los agentes no necesariamente deben conocerse entre sí. La principal forma de comunicación es a través del paso de mensajes. Los mensajes son objetos. Un objeto mensaje está caracterizado por `kind`, una propiedad que es usada para distinguir los mensajes. La clase message soporta un rango de constructores que tienen `kinds` como argumento [Lange y Oshima, 1998].

3 MARCO APLICATIVO

3.2 INTRODUCCIÓN

En el presente capítulo se dará seguimiento al desarrollo de la programación y gestación del prototipo requerido que pueda contener las características necesarias para demostrar que los procesos de votación electrónica pueden optimizar en muchos aspectos cualquier tipo de acto electoral, el desarrollo de esta plataforma pretende brindar una posible solución, además de ser un gran paso al progreso de la tecnología en nuestro país, en un campo tan delicado como lo es el trato de la democracia.

3.3 ANÁLISIS DE REQUERIMIENTOS

El desarrollo de la arquitectura del sistema surge del reconocimiento de las necesidades que este deba cumplir para poder ayudar a la demostración de la hipótesis. Cada uno de los requerimientos que se puedan identificar en el proceso de análisis ayudará a la realización del diseño del sistema, dándonos parámetros los cuales se convertirán en los pilares de la arquitectura del mismo y sobre los cuales se construirá el prototipo

(software). Estos requerimientos se dividen en dos categorías, los funcionales y los no funcionales. Los funcionales son aquellos que especifican el funcionamiento del sistema, tales como las restricciones, limitaciones, servicios y funciones ofrecidas. Por otro lado, los no funcionales son los aspectos del sistema visibles para el usuario que no están relacionados de forma directa con el comportamiento funcional del sistema.

3.3.1 Visión del Sistema

En la actualidad las personas han ido avanzando junto con la tecnología, relacionando y utilizando esta para realizar diversas actividades, las cuales están relacionadas a un sinnúmero de rubros, pero algo con lo cual se ha tropezado es con los procesos electorales, como se puede ver en muchos países, por más desarrollados que estos estén, se siguen utilizando papeletas impresas, ánforas y otros los cuales deberían ser reemplazados por software y hardware que cumpla los requerimientos que las personas necesiten en este campo. Y una de las principales susceptibilidades por las cuales no es tan aceptado este salto tecnológico de este proceso es la seguridad, pues con el avance de las diferentes tecnologías en hardware y software, también a la par han evolucionado las técnicas y conocimientos para la vulneración de estos.

Por lo tanto se propone desarrollar un sistema que pueda servir como un plataforma virtual de votación, diseñada para su funcionamiento en una red LAN o en un INTRANET, pero podría ser fácilmente adaptada para realizar algunos procesos en INTERNET, esta característica coadyuvaría con la seguridad, brindando a las personas la confianza necesaria para realizar la votación sabiendo que no será mal utilizado su derecho a elegir. El sistema estará basado en agentes móviles, los cuales podrán movilizarse por la red, así como comunicarse con otros agentes especializados, esta comunicación permitirá la exclusión del sistema del peligro de una intrusión

externa a través de INTERNET. Se podría analizar una secuencia de procesos y acciones a realizarse durante la ejecución del sistema.

1. El votante pedirá realizar la votación
2. El administrador de la mesa de sufragio llenara el formulario de validación del votante.
3. Finalizado el llenado el administrador dejara que el agente después de verificar los datos realice una petición a un agente en el nodo cliente, para la activación de la papeleta virtual.
4. El agente en el nodo de cliente se activara y mostrara la boleta de sufragio virtual.
5. Después de realizar su elección y confirmar su voto el agente cliente se comunicara con el agente del nodo administrador, llevando los datos de la votación.
6. Se realizara el registro de la información de la votación así como del registro del votante
7. En cualquier momento se puede ejecutar el agente del servidor, el cual enviara clones a los otros nodos administradores, recolectando información vital del proceso y realizando la centralización de la información.

Este proceso se puede advertir más claramente en el grafico (Figura 3.1).

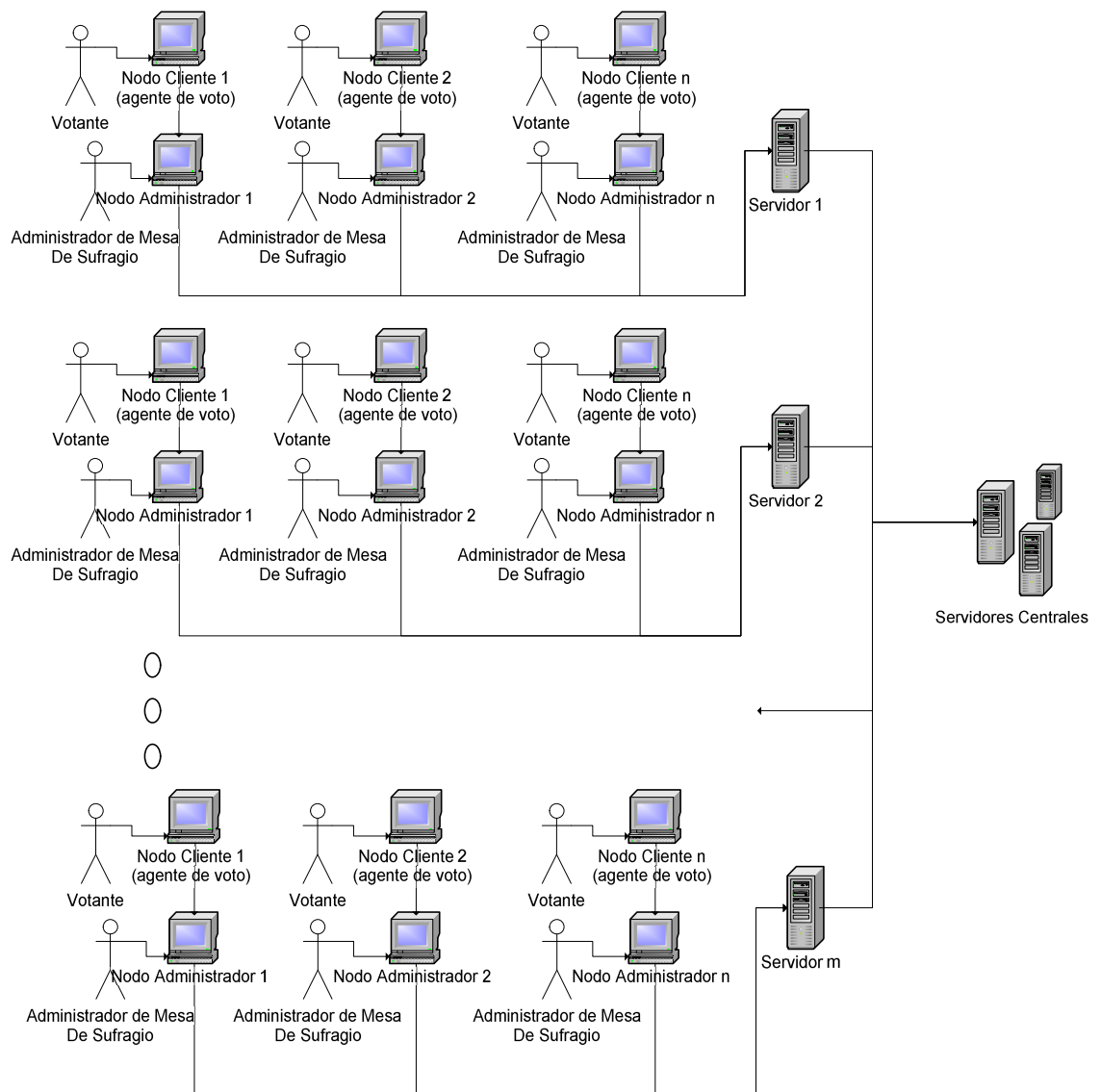


Figura 3.1.- Red de distribución de actores. Fuente: [Elaboración Propia]

3.3.2 Requerimientos Funcionales

Como se menciona antes, los requerimientos funcionales son las restricciones limitaciones, servicios y funciones ofrecidas por el sistema. Por medio de ellos se podrán identificar y definir una serie de elementos que van a componer la arquitectura con el fin de cumplir las funcionalidades requeridas.

Después de realizar el análisis correspondiente se pudieron identificar cuatro actores: el votante, el administrador de la mesa de sufragio, el

administrador del servidor secundario, administrador del servidor primario. El votante es la persona que después de brindar sus datos para la validación de los mismos se dirige al terminal donde se le mostrará una papeleta virtual, en la cual marcará su elección y confirmará la misma, en ese mismo instante un agente se comunicara con el nodo del administrador de la mesa y viajara con la información del voto respectivo, después el agente del nodo del administrador, se encargará de procesar la información del voto. El administrador de la mesa de sufragio es la persona la cual tendrá la tarea de rellenar el formulario de validación del votante, inmediatamente se activara el agente, que enviará un mensaje al agente del nodo cliente para que se active la papeleta virtual de sufragio. Los otros dos administradores se encargaran de realizar los cambios de itinerarios en las tareas de los agentes de centralización situados en los servidores, además podrán monitorear los resultados parciales y observar el comportamiento de los datos. Se puede ver más claramente en el diagrama de casos de uso de los mismos (Figura 3.2).

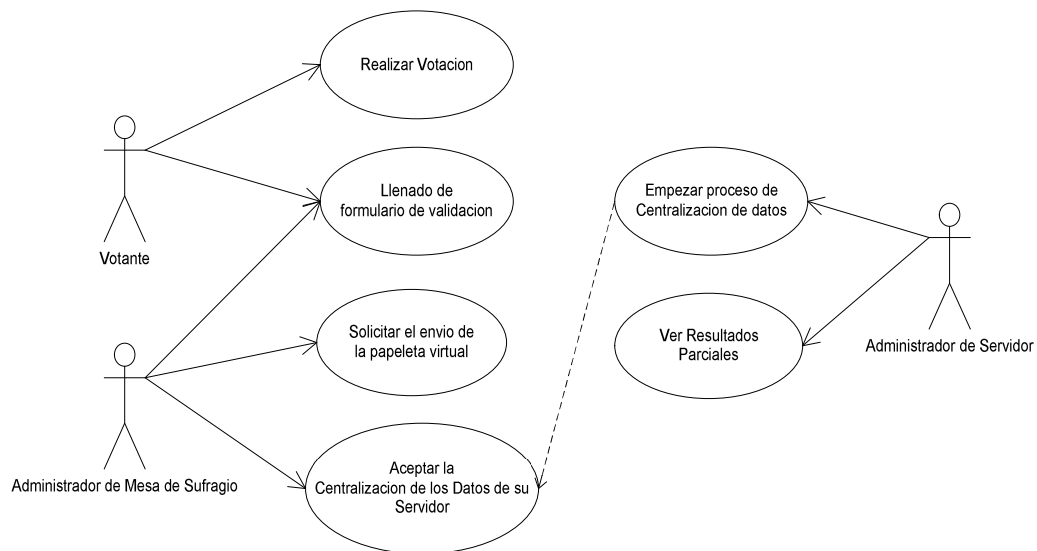


Figura 3.2 Casos de uso de los actores humanos. Fuente: [Elaboración Propia]

Además de los usuarios como personas tenemos que hablar de los agentes como parte de este sistema como un tipo especial de usuarios ya que

estarán encargados de realizar tareas tan complejas como las de los usuarios humanos (Figura 3.3).

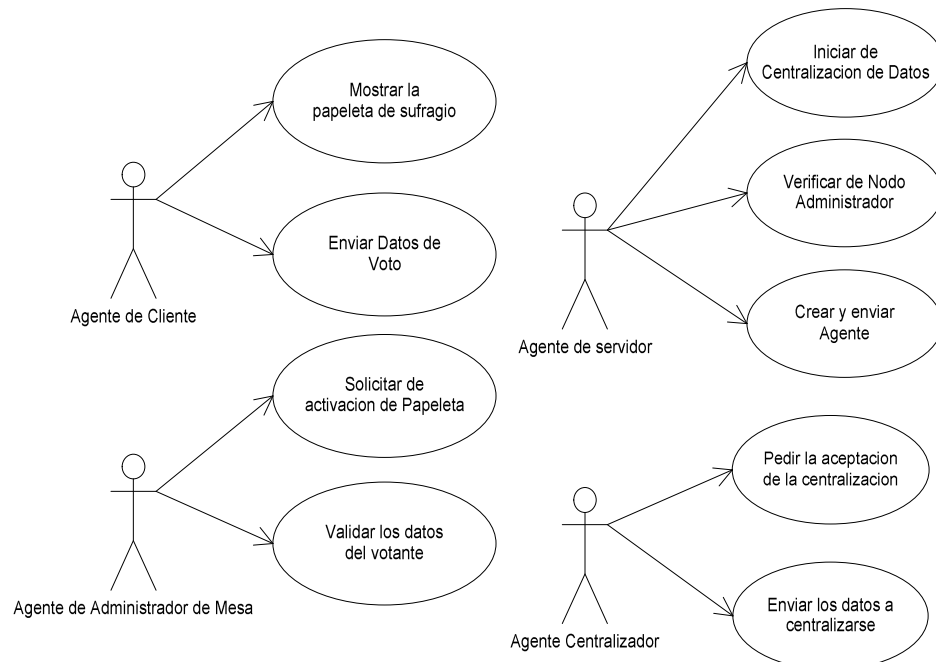


Figura 3.3 Casos de uso de los actores Agentes. Fuente: [Elaboración Propia]

3.3.3 Requerimientos No Funcionales

Se definió a los requerimientos no funcionales como los aspectos del sistema visibles para el usuario que no están relacionados de forma directa con el comportamiento funcional del sistema. Los requerimientos no funcionales al igual que los requerimientos funcionales, ayudan a identificar elementos que a componer la arquitectura. Estos componentes tienen la finalidad de ayudar a que la funcionalidad establecida en los casos de uso pueda ser llevada a cabo de manera natural.

Después del análisis se pudo identificar los siguientes requerimientos no funcionales:

1. Comunicación entre agentes
2. Ejecución en paralelo
3. Tolerancia a fallos
4. Concurrencia

5. Escalabilidad
6. Distribuido
7. Tiempo al aire
8. Conexiones asíncronas

3.3.3.1 Comunicación entre Agentes

Los agentes deberán poder comunicarse entre sí, con el fin de poder ejecutar las tareas solicitadas. Cuando un agente llegue a una plataforma, debe poder comunicarse con el agente que le presta el servicio (si es que existe uno ahí) y comunicarle los datos que le proporcionó el usuario. Así mismo el agente residente en el servidor deberá comunicarse con el agente recibido para enviarle la respuesta.

3.3.3.2 Ejecución en Paralelo

Varios agentes podrán ser ejecutados en paralelo dentro de un servidor que provea servicios a los agentes. Esto quiere decir, que los agentes no tendrán que esperar en una cola mientras cada uno van siendo ejecutados uno por uno, sino que muchos podrán ser ejecutados al mismo tiempo con el fin de evitar demoras.

3.3.3.3 Tolerancia a fallos

El sistema deberá poder persistir a los agentes enviados por los usuarios para realizar las tareas solicitadas, con el fin de evitar la pérdida de agente y la información que le fue proporcionada en caso de algún error. Los agentes deberán ser persistidos después de ejecutar una tarea, así para evitar perder el resultado de la ejecución de la tarea.

3.3.3.4 Concurrencia

El sistema debe soportar una gran cantidad de agentes entrando, saliendo y ejecutando servicios al mismo tiempo. Para ello deberá optimizar recursos al máximo con el fin de evitar una falla en caso de sobrecarga.

3.3.3.5 Escalabilidad

El sistema deberá poder soportar el crecimiento del número de usuarios a través del tiempo, sin tener un gran impacto en los tiempos de respuesta. Para ellos deberá utilizar colas y optimizar recursos, al igual que proveer una serie de dispositivos de apoyo.

3.3.3.6 Distribuido

El sistema deberá ser distribuido con el fin de cumplir los requerimientos de escalabilidad y concurrencia. Al tenerlo distribuido, se están evitando los cuellos de botella y la dependencia específica sobre algún componente.

3.3.3.7 Tiempo al aire

El sistema debe reducir al máximo el tiempo al aire que deben estar conectados los dispositivos de los usuarios.

3.3.3.8 Conexiones Asíncronas

El sistema debe soportar conexiones de tipo asíncronas con el fin de reducir el tiempo al aire. Es decir, el usuario mandará al agente a que ejecute el servicio y se podrá desconectar ahí mismo sin tener que esperar la respuesta. Luego se volverá a conectar para pedir la respuesta al servicio solicitado.

3.4 MODELO DE AGENTES

Continuando con el desarrollo del prototipo se procederá a especificar los agentes móviles que intervienen en la aplicación. Cada uno de estos agentes tendrá un comportamiento característico que permitirá realizar la votación por medio de una papeleta virtual la cual está controlada por un agente móvil que viajara hasta el host indicado, así como de otros agentes

encargados de realizar el registro y centralización de la información de los resultados parciales y totales.

Este capítulo estará constituido por tres secciones. En la primera se explicara una metodología orientada a agentes. En la segunda se abordara en detalle al modelado de los agentes y por último se estudiaran los tipos de agentes que intervendrán en los procesos de la plataforma.

3.4.1 Metodología AOPOA

La Metodología AOPOA (Aproximación Organizacional para Programación Orientada a Agentes) [Ahogado & Reinemer, 2003] consta de dos etapas principales, análisis y diseño organizacional. Son aplicadas de manera iterativa, descomponiendo lo objetos y roles que se han identificado hasta determinar que ya no es necesario este proceso. Una vez efectuado lo anterior, se procede a realizar una última iteración donde se efectúa el diseño de los agentes, especificando cuales son necesarios para cumplir con los objetivos planteados y la forma de cumplirlo. Como paso final viene todo el proceso de implementación del diseño. A continuación se describirán las etapas con sus pasos intermedios:

Análisis:

- Caracterizar el sistema.
- Definir enfoque OA (orientado a agentes)
- Evaluar complejidad
- Encontrar roles

Diseño Organizacional:

- Caracterizar interacciones
- Manejar interacciones
- Aplicar protocolos de interacción
- Identificar vínculos de comunicación

Diseño de agentes:

- Especificar los tipos de agentes
- Establecer arquitectura interna de cada agente
- Implementación

- Establecer herramienta
- Implementar
- Realizar pruebas

3.5 ESPECIFICACIÓN DE LOS AGENTES

3.5.1 Agente Servidor de Centralización (MasterC) (Primera Iteración)

Agente ServidorC

a. Estado

- i. Interfaz de ingreso de datos de para servidores de destino
- ii. Interfaz de retorno de información de situación actual en la votación

b. Metas

M11. Obtener las direcciones de los servidores intermedios

M12. Enviar información a un Agente Centralizador (SlaveC)

M13. Obtener la información del Agente Centralizador.

M14. Presentar los datos centralizados al usuario

c. Entradas Sensoriales

i. Asíncronas

S11. Solicitud de ejecución por parte del usuario

S12. Datos de retorno del Agente Centralizador

ii. Síncronas

Ninguna

d. Actuadores

A11. Crear el Agente Centralizador

A12. Enviar al Agente Centralizador a un destino del itinerario

e. Comportamientos

B11. Crear Agente Centralizador: cumple con M12.

Mensaje/Evento: Enviar al Agente Centralizador a una dirección específica por medio de S11.

Acción: Desplegar una interfaz grafica para obtener la dirección del servidor al que se enviara el agente esclavo mediante S11. En caso que la respuesta sea afirmativa se creara un Agente Centralizador.

Respuesta: Confirmación de la creación y envío del agente esclavo.

B12. Presentar la Centralización de los datos y los resultados cumpliendo con M13 y M14.

Mensaje/Evento: Muestra en las casillas correspondientes los datos de los resultados totales por medio de S12. En caso afirmativo muestra los resultados totales

Respuesta: Ninguna.

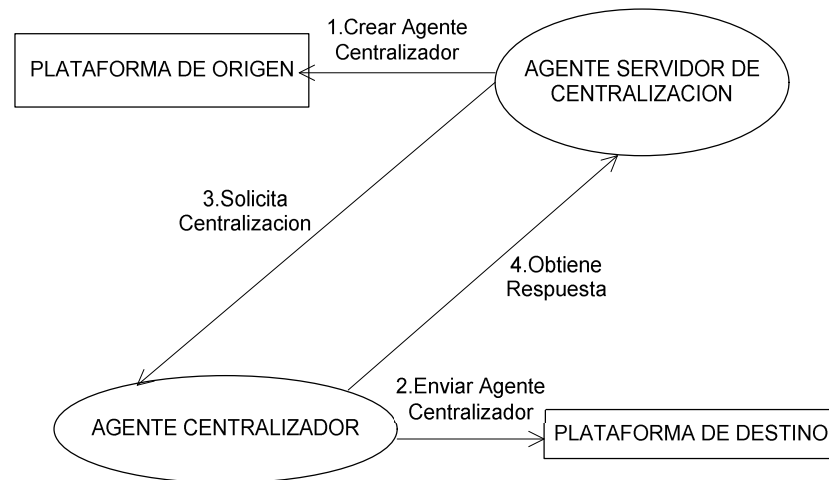


Figura 3.4 Diagrama del agente Servidor de Centralización. Fuente: [Elaboración Propia]

3.5.2 Agente Centralizador (SlaveC) (Primera Iteración)

Agente EsclavoC

a. Estado

- i. Interfaz de retorno de información de situación actual en la votación

b. Metas

M11. Obtener los datos del host de destino

M12. Enviar información a un Agente Servidor de Centralización

c. Entradas Sensoriales

- i. Asíncronas

S11. Datos de retorno del Agente Centralizador

- ii. Síncronas

Ninguna

d. Actuadores

A11. Enviar al Agente Centralizador a un destino del itinerario

e. Comportamientos

B11. Enviar información de resultados parciales: cumple con M12.

Mensaje/Evento: Enviar al Agente Servidor de Centralización la información de la votación parcial.

Acción: Desplegar una interfaz grafica para obtener aceptación del proceso de centralización mediante S11. En caso que la respuesta sea afirmativa se enviara la información de la votación.

Respuesta: ninguna

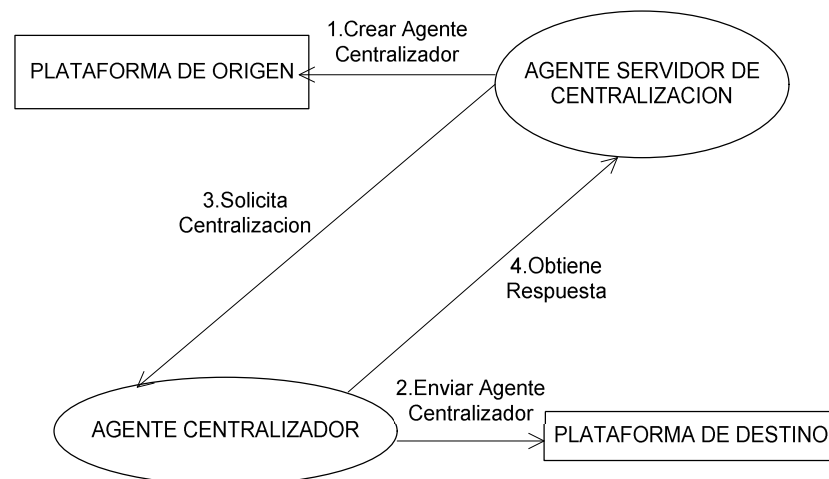


Figura 3.5 Diagrama del Agente Centralizador. Fuente: [Elaboración Propia]

3.5.3 Agente Servidor de Votación (MasterServer) (Segunda Iteración)

Agente ServidorMaestro

a. Estado

- i. Interfaz de ingreso de datos de para validación de datos
- ii. Interfaz de retorno de información del votante

b. Metas

M11. Obtener los datos del votante para permitir la votación

M12. Enviar a un Agente de Usuario (SlaveUser)

M13. Obtener la información del Agente de Usuario.

M14. Realizar el almacenamiento del voto emitido

c. Entradas Sensoriales

i. Asíncronas

S11. Solicitud de ejecución por parte del usuario

S12. Datos de retorno del Votante

ii. Síncronas

Ninguna

d. Actuadores

A11. Crear el Agente de Usuario

A12. Enviar al Agente de Usuario al destino del votante

e. Comportamientos

B11. Crear Agente de Usuario: cumple con M12.

Mensaje/Evento: Enviar al Agente de Usuario a una dirección específica por medio de S11.

Acción: Desplegar una interfaz grafica para obtener la información del votante para confirmar el envío del Agente de Usuario mediante S11. En caso que la respuesta sea afirmativa se creara un Agente de Usuario.

Respuesta: Confirmación de la creación y envío del agente esclavo.

B12. Presentar la información de los datos y los resultados cumpliendo con M13 y M14.

Mensaje/Evento: Recibe el mensaje con el voto realizado y se almacena por medio de S12.

Respuesta: Ninguna.

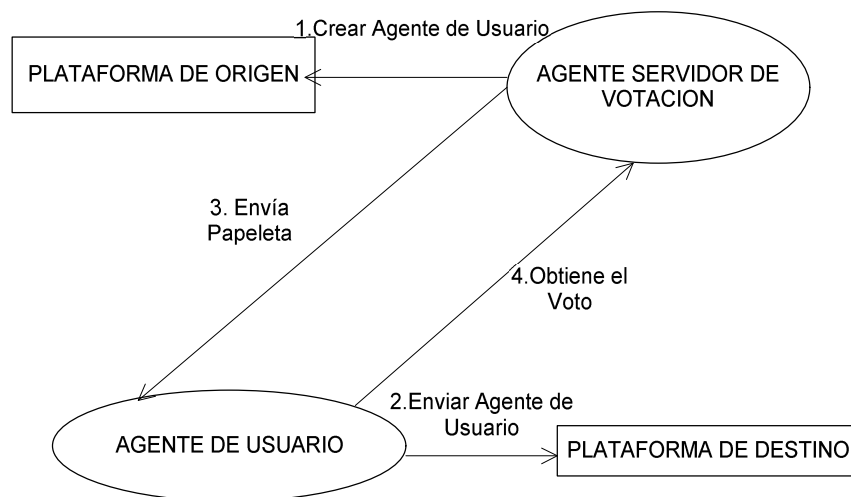


Figura 3.6 Diagrama del agente Servidor de Votación

3.5.4 Agente de Usuario (SlaveUser) (Segunda Iteración)

Agente Usuario Esclavo

a. Estado

i. Interfaz de retorno de información del voto

b. Metas

M11. Obtener la información del Voto.

M12. Enviar la información del voto del usuario

c. Entradas Sensoriales

i. Asíncronas

S11. Datos del Voto del Usuario

ii. Síncronas

Ninguna

d. Actuadores

A11. Enviar un mensaje con la información de la elección del usuario

e. Comportamientos

B11. Crear una papeleta de votación: cumple con M12.

Mensaje/Evento: Enviar información del voto por medio de S11.

Acción: Desplegar una interfaz grafica para obtener la elección del usuario mediante S11. Después se envía el dato al Servidor de Votación para su posterior almacenamiento

Respuesta: ninguna.

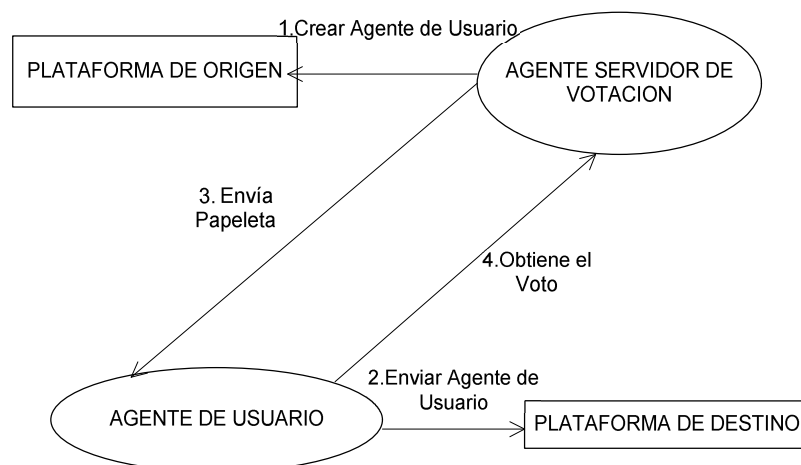


Figura 3.7 Diagrama del Agente de Usuario. Fuente: [Elaboración Propia]

3.6 DISEÑO DE LA PLATAFORMA

La plataforma virtual basada en agentes para comicios electorales es un sistema distribuido que cuenta con varios agentes distribuidos en varios servidores, los cuales están localizados físicamente alejados unos de otros, aunque las distancias entre estos pueden variar, el esquema de la Figura 3.1 se cumple a cabalidad, para proporcionar estabilidad a la jerarquía presentada y el desenvolvimiento de todas las partes que componen los diversos actores que interactúan con el sistema.

En esta parte del desarrollo de la creación del prototipo se analizará tres aspectos importantes. El primer aspecto a analizar será el diagrama de capas de la plataforma. El segundo aspecto será el análisis a los módulos que componen el sistema y el tercer aspecto consiste en mostrar en detalle cada uno de los módulos que componen el sistema mostrando los diagramas de clases y de secuencia.

3.6.1 Diagrama de Capas

El diagrama de capas ilustra la arquitectura de la plataforma. Teniendo en cuenta cada uno de los dispositivos donde se implementará la plataforma Figura 3.8.

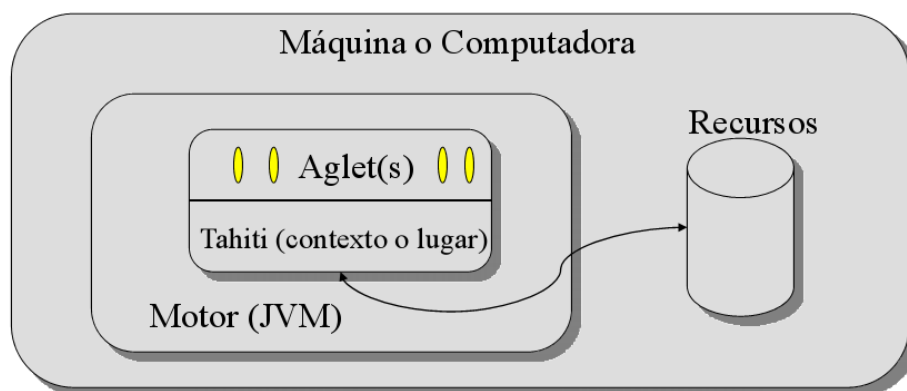


Figura 3.8 Entorno de Funcionamiento de un aglet [Yágüez J, 2010]

En esta figura se puede observar la estructura en la cual se desenvuelve un aglet, y se puede observar como JVM (Java Virtual Machine) y el servidor Tahiti proporcionan juntos el acceso controlado a los recursos y servicios locales.

3.6.2 Especificación de los casos de uso

Como hemos podido definir se encontraron los casos de uso de cada uno de los actores correspondientes a la estructura de votación que se desea implementar para el funcionamiento de la plataforma en su entorno.

A continuación con ayuda de los casos de uso identificados en las figuras (Figura 3.2 y Figura 3.3), se procederá a detallar estos casos de uso para la posterior verificación del cumplimiento de los requerimientos necesarios en el desarrollo de la plataforma.

Caso de uso:	Realizar Votación
Actor:	Votante
Propósito:	Elegir una opción predeterminada
CURSO NORMAL DE EVENTOS	
Acción de los actores	
<ol style="list-style-type: none"> 1. El caso de uso se inicia después de haber verificado la información personal en la mesa de sufragio. 2. El votante pasa al recinto donde se encuentra un terminal donde se ha recibido la boleta de sufragio. 3. El votante selecciona una opción y acepta la emisión de su voto. 	

Caso de uso:	Llenado del formulario de validación
Actor:	Votante y Administrador de mesa de sufragio
Propósito:	Validar los datos del votante para permitirle emitir su voto
CURSO NORMAL DE EVENTOS	

Acción de los actores
<ol style="list-style-type: none"> 1. El Votante se apersona a la mesa de sufragio y brinda su información personal (CI) con su documento original. 2. El Administrador de la mesa de sufragio llena el formulario de validación y solicita la validación de los mismos.

Caso de uso:	Solicitar el envío de la papeleta virtual
Actor:	Administrador de mesa de sufragio
Propósito:	Permitir al votante que realice su votación
CURSO NORMAL DE EVENTOS	
Acción de los actores	
<ol style="list-style-type: none"> 1. Después de haber validado la información del votante, el administrador de la mesa solicita el envío de la papeleta virtual. 	

Caso de uso:	Aceptar la centralización de los datos de su servidor
Actor:	Administrador de la mesa de sufragio
Propósito:	Permitir el acceso a los datos de la votación en la mesa
CURSO NORMAL DE EVENTOS	
Acción de los actores	
<ol style="list-style-type: none"> 1. El administrador de la mesa recibe un mensaje que indica que se centralizaran los datos de su mesa, al aceptar, los datos con enviados a través del agente centralizador. 	

Caso de uso:	Empezar proceso de centralización
Actor:	Administrador de Servidor
Propósito:	Obtener la información de la votación de las mesas de sufragio para obtención de resultados centralizados
CURSO NORMAL DE EVENTOS	
Acción de los actores	
<ol style="list-style-type: none"> 1. El administrador introduce la dirección del servidor que se encuentra en cada mesa. 2. El administrador permite el envío del agente centralizador. 	

Caso de uso:	Ver resultados Parciales
Actor:	Administrador de servidor
Propósito:	Poder observar los resultados actuales después de haber centralizado los datos de cierto número de mesas
CURSO NORMAL DE EVENTOS	
Acción de los actores	
<ol style="list-style-type: none"> 1. El administrador solicita mostrar los datos actuales centralizados en su terminal presionando el botón centralizar. 2. El sistema muestra los datos numéricos de la votación . 	

CURSO DE LOS EVENTOS SEGÚN LOS CASOS DE USO U LA RESPUESTA DEL SISTEMA

Caso de uso:	Iniciar sesión
Actor:	usuario (iniciador)

Propósito:	Iniciar sesión
CURSO NORMAL DE EVENTOS	
Acción de los actores	Respuesta del sistema
1. El caso de uso inicia cuando el usuario desea acceder al sistema.	2. Petición de identificador y contraseña del usuario.
3. Registra su correspondiente identificador y contraseña.	5. Despliegue de la pantalla principal del sistema, de acuerdo a los privilegios que tiene el usuario.
4. Presiona el botón "login"	6. fin

Caso de uso:	Realizar votación
Actor:	Administrador de mesa, Votante, plataforma de votación
Propósito:	Cotejar los pasos necesarios y el proceso que se debe seguir
CURSO NORMAL DE EVENTOS	
Acción de los actores	Respuesta del sistema
1. El caso se inicia cuando el votante se apersona a la mesa de sufragio y solicita realizar la votación.	
2. Para ello, el administrador ingresa al sistema su número de CI	4. Despliega la interfaz grafica para realizar validación, así como muestra al administrador de la mesa si el votante está habilitado.
3. selecciona "Validar" para mostrar los datos y el estado	6. El sistema envía la papeleta virtual al host del recinto que desplegara la ventana de la

del votante.	papeleta de votación.
5. Si 4 es verdadero, el administrador de la mesa presiona la tecla “enviar”, para iniciar con la migración del agente esclavo.	8. El agente esclavo, manda los datos de la elección del elector.
7. El votante realiza su elección.	9. El agente servidor de la mesa, recibe la información de la votación y la almacena, además de desplegar un mensaje de confirmación de la llegada del voto
8. Finaliza las acciones de este caso de uso.	

Caso de uso:	Centralizar los datos de los servidores
Actor:	Administrador de servidor, administrador de mesa y la plataforma virtual
Propósito:	Realizar el traspaso de la información de la votación desde los servidores de la mesas de sufragio a servidores centralizadores
CURSO NORMAL DE EVENTOS	
Acción de los actores	Respuesta del sistema
1. El caso de uso inicia cuando el administrador de los servidores centralizadores requiere obtener los datos parciales o totales de las respectivas mesas.	4. El sistema envía un agente centralizador a los destinos y el agente centralizador se pone

<p>2. Para ello, el administrador ingresa a la interfaz de centralización de datos.</p> <p>3. introduce la dirección de los servidores y luego presiona la opción de “enviar”</p> <p>5 En la interfaz del servidor de mesa se pide confirmación de la centralización de los datos.</p> <p>6. Una vez que se enviaron los datos desde las mesas, el administrador de los servidores centralizadores puede presionar “Centralizar”, para observar los datos actualizados en pantalla</p> <p>8. El administrador visualiza los datos.</p> <p>9. Finaliza las acciones de este caso de uso.</p>	<p>en espera de los datos enviados por los agentes centralizadores.</p> <p>7. El sistema realiza la operación solicitada, mostrando los datos recibidos de los agentes centralizadores.</p>
---	---

CASOS DE USO DE LOS AGENTES

Caso de uso:	Mostrar papeleta de sufragio
Actor:	Agente(Agente de Cliente)

Propósito:	Permitir al votante realizar la votación mostrando la ventana de opciones
CURSO NORMAL DE EVENTOS	
Acción de los actores	
<ol style="list-style-type: none"> 1. El agente viaja a pedido de un servidor que se encuentra en la mesa de sufragio. 2. El agente muestra la ventana con las opciones para la votación. 	

Caso de uso:	Enviar datos del voto
Actor:	Agente (Agente de cliente)
Propósito:	Permitir a la mesa registrar la opción seleccionada por el votante
CURSO NORMAL DE EVENTOS	
Acción de los actores	
<ol style="list-style-type: none"> 1. El agente después de haber recibido la opción seleccionada por el votante envía la información al servidor de la mesa de sufragio. 	

Caso de uso:	Solicitar la activación de la papeleta
Actor:	Agente de administrador de mesa
Propósito:	Crear y enviar un agente de cliente que permita obtener los datos del voto
CURSO NORMAL DE EVENTOS	
Acción de los actores	

1. El agente administrador de la mesa de sufragio después de validar los datos del votante y verificar que este habilitado , crea y envía un agente de cliente.

Caso de uso:	Validar los datos del votante
Actor:	Agente de administrador de mesa, administrador de mesa
Propósito:	Determinar si el votante está habilitado para realizar la votación
CURSO NORMAL DE EVENTOS	
Acción de los actores	
<ol style="list-style-type: none"> 1. El administrador de mesa llena el formulario de validación y luego solicita la validación. 2. El agente de administrador de mesa ingresa a los datos y los muestra en una ventana con la verificación habilitación del votante 	

Caso de uso:	Iniciar Centralización Datos
Actor:	Agente de servidor; Administrador de mesa
Propósito:	Centralizar los datos desde los servidores de las mesas de sufragio
CURSO NORMAL DE EVENTOS	
Acción de los actores	
<ol style="list-style-type: none"> 1. El administrador de la mesa de sufragio solicita la centralización de los datos de un determinado servidor que se encuentra en la mesa de sufragio. 2. El agente servidor crea un Agente Centralizador. 3. El agente servidor envía al Agente Centralizador. 	

Caso de uso:	Verificar Nodo
Actor:	Agente Servidor
Propósito:	Iniciar comunicación para el envío de un Agente Centralizador
CURSO NORMAL DE EVENTOS	
Acción de los actores	
<ol style="list-style-type: none"> 1. El agente servidor verifica y obtiene la dirección del servidor al que se enviará el Agente Centralizador. 	

Caso de uso:	Crear y enviar Agente Centralizador
Actor:	Agente Servidor , administrador de servidor
Propósito:	Crear enviar un Agente Centralizador a una dirección específica que permita la centralización de los datos de este.
CURSO NORMAL DE EVENTOS	
Acción de los actores	
<ol style="list-style-type: none"> 1. El administrador de servidor, selecciona una dirección de un servidor de una mesa de sufragio y solicita el envío de un agente centralizador. 2. El agente servidor crea un Agente Centralizador. 3. El agente servidor envía el Agente Centralizador creado 	

Caso de uso:	Pedir la aceptación de la centralización
Actor:	Agente centralizador
Propósito:	Avisar al administrador de la mesa de sufragio que los

	datos de su servidor serán centralizados.
CURSO NORMAL DE EVENTOS	
Acción de los actores	
<ol style="list-style-type: none"> 1. Después de llegar al servidor de destino el agente Centralizador pide una confirmación de la centralización de los datos de su servidor de mesa. 2. Muestra una ventana indicando que los datos serán centralizados. 	

Caso de uso:	Enviar los datos a centralizarse
Actor:	Agente Centralizador , administrador de mesa
Propósito:	Enviar los datos necesarios a los servidores que centralizaran la información de cada mesa de sufragio
CURSO NORMAL DE EVENTOS	
Acción de los actores	
<ol style="list-style-type: none"> 1. El administrador después de haber recibido la petición de centralización acepta. 2. El agente centralizador accede a los datos del servidor de la mesa de sufragio. 3. El agente centralizador envía los datos al servidor de centralización. 	

3.6.3 Especificación del Sistema

La arquitectura de la plataforma está dividida en dos módulos que se encargan del manejo del proceso de votación en la mesa de sufragio y de la centralización de la información de los resultados de la votación también se puede observar las clases que intervienen en la plataforma y los métodos que estos utilizan para poder cumplir el fin asignado Figura 3.9.

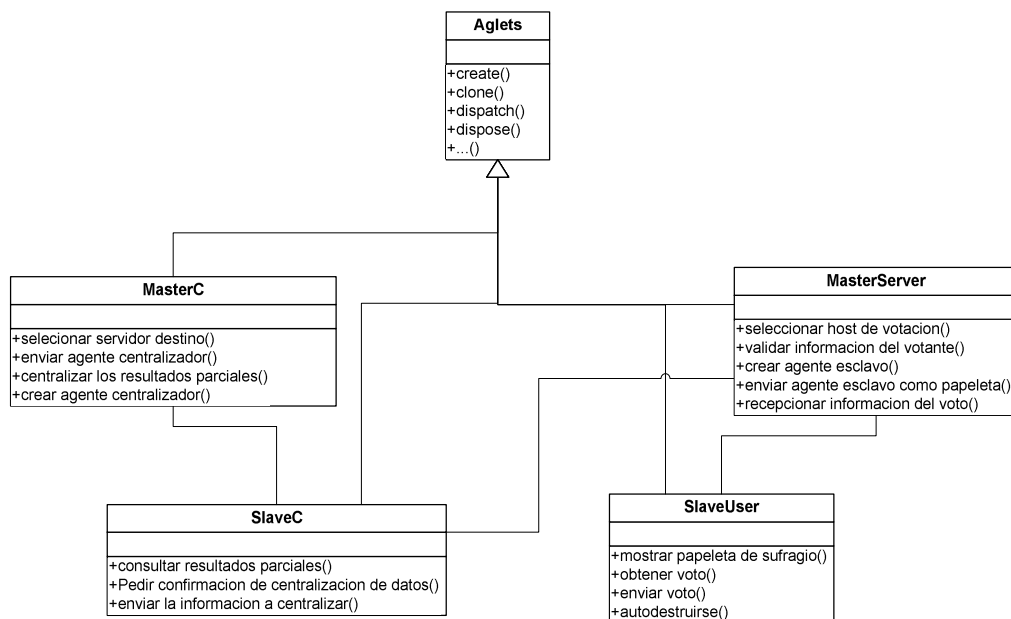


Figura 3.9 Diagrama de clases de la plataforma. Fuente: [Elaboración Propia]

3.6.4 Diagramas de Secuencia

Los diagramas de secuencia son imprescindibles en la etapa de análisis, toda vez que contribuyen de manera importante a entender el comportamiento del sistema.

Cada diagrama de secuencia, representa un determinado escenario de un caso de uso, que tuvieron que ser trabajadas en el curso normal de eventos, perteneciente a la etapa de la obtención de requisitos.

3.6.4.1 Diagrama de Secuencia de Centralización de datos

En este diagrama de secuencia se explica los procedimientos que intervienen en el proceso de centralización de los datos alojados en los servidores de las mesas de sufragio.

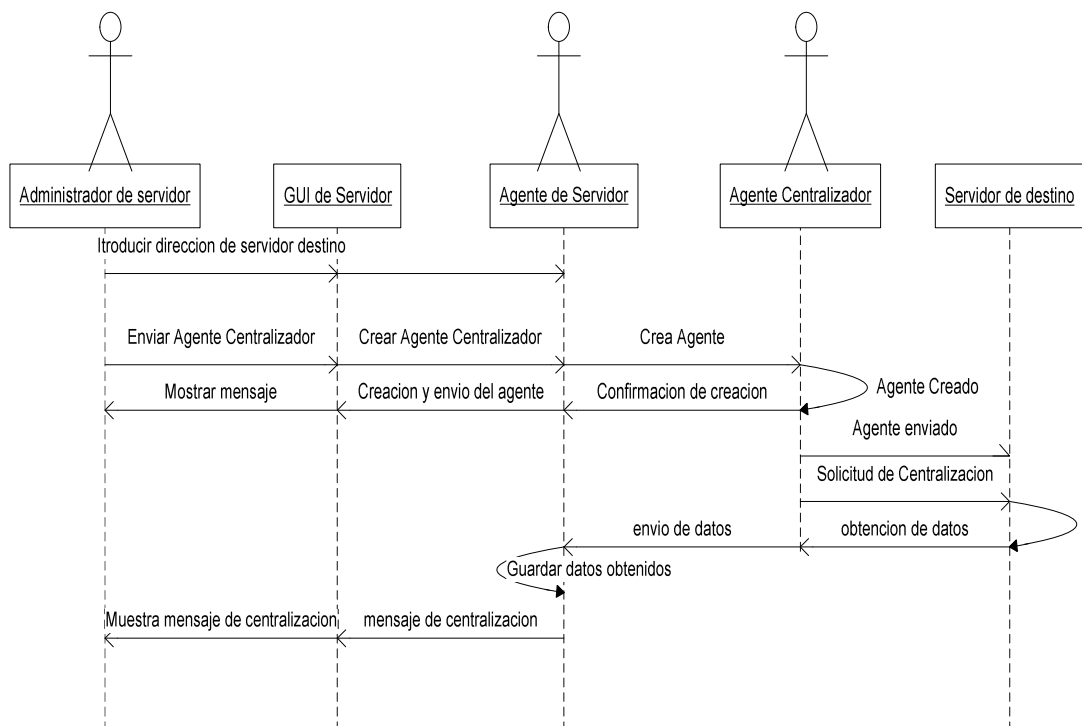


Figura 3.10 Diagrama de secuencia de la centralización de datos. Fuente: [Elaboración Propia]

3.6.4.2 Diagrama de Secuencia de Mostrar Resultados

El diagrama de secuencia brinda los pasos que siguen los procesos cuando uno de los administradores de los servidores desea ver los resultados, ya sean parciales o totales.

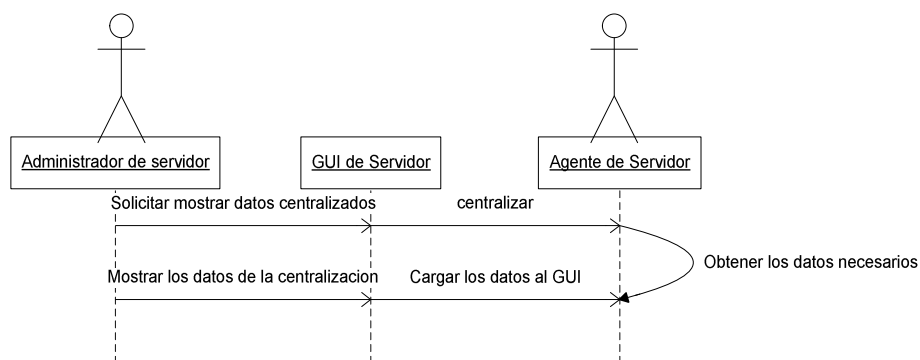


Figura 3.11 Diagrama de secuencia de mostrar los datos centralizados. Fuente: [Elaboración Propia]

3.6.4.3 Diagrama de Secuencia de Emisión de Voto

En el presente diagrama de secuencia se observa el proceso que sigue la información y los métodos que utilizan los actores para poder realizar la votación, se puede explicar este proceso, como algo parecido a lo que sucede en la actualidad, un votante se apersona a su mesa de sufragio, luego de verificar que el votante está habilitado para votar y sus datos corresponde a esa misma persona se le brinda la papeleta, que en este caso se enviara a la computadora por medio de un agente que luego de obtener su voto lo enviara para que se contabilice en el servidor de la mesa de sufragio.

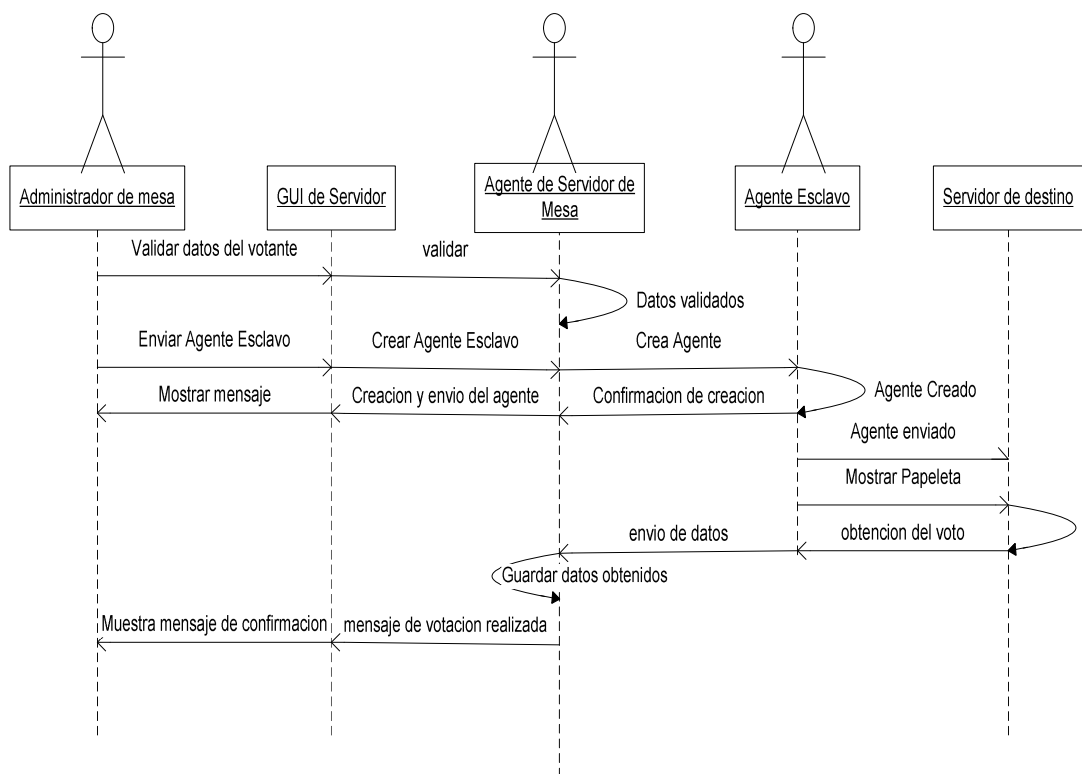


Figura 3.12 Diagrama de secuencia de la emisión del voto. Fuente: [Elaboración Propia]

3.6.4.4 Diagrama de Secuencia de Centralización de datos

En este diagrama de secuencia se puede observar las acciones y métodos que intervienen en el proceso de validación de la información del votante, esto sucede cuando un votante se apersona a una de las mesas de sufragio para realizar la votación y uno de los delegados en la mesa de sufragio debe pedir su documento de identidad para poder verificar que el votante este habilitado para sufragar.

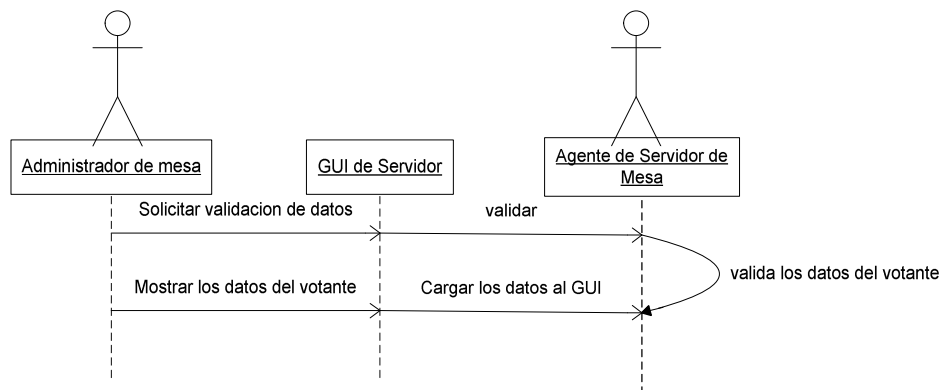


Figura 3.13 Diagrama de Secuencia de Validación de datos del votante.
Fuente: [Elaboración Propia]

3.6.5 Desarrollo de Interfaces Graficas

Las interfaces graficas permiten a los usuarios tener entornos amigables y fáciles de entender , pues las personas que serán designadas como administradores de mesa, tendrán un conocimiento básico del manejo de una computadora, eso nos obliga a que las acciones que se realice para el proceso de desarrollo de las interfaces graficas tienen que ser sencillas.

3.6.5.1 Interfaz de Inicialización del Servidor Tahití

La interfaz del servidor que se utilizara es el Tahití, el cual es proporcionado por Aglets y las librerías descargadas desde su página principal. Este

entorno aparece luego de la inicialización del servidor, donde nos pedirá una contraseña (Figura 3.14).



Figura 3.14 Ventana de Ingreso de Usuario del servidor Tahití. Fuente: [Elaboración Propia]

Luego de ingresar los datos correctos para la inicialización del servidor nos mostrara el panel principal de este (Figura 3.15).

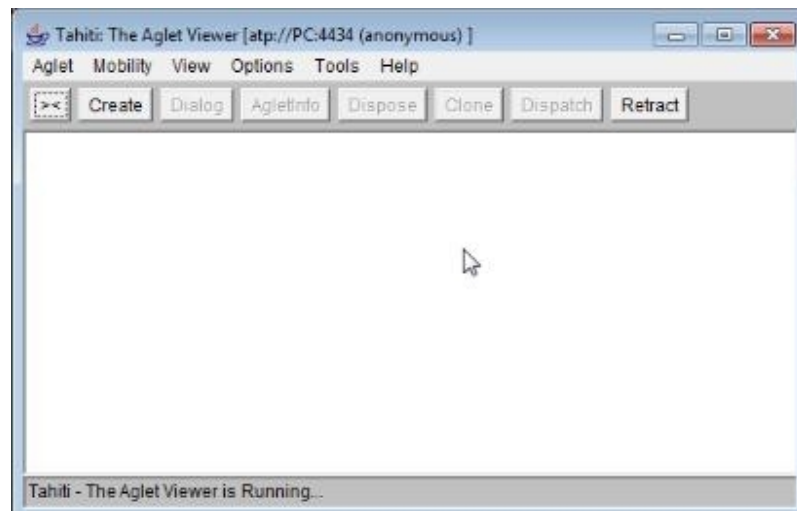


Figura 3.15 Ventana Principal del Servidor Tahití. Fuente: [Elaboración Propia]

Desde esta ventana se puede controlar y observar el comportamiento de los Aglets que corren sobre este servidor, es sobre el mismo que se desarrollara el funcionamiento de la plataforma, este es el que nos permitirá usar las características propias de los agentes móviles.

3.6.5.2 Interfaz del Agente Servidor de la Mesa de Sufragio

Para poder simplificar las tareas de las cuales tiene que hacerse cargo la persona que administre este servidor, se concentrará en la simplicidad del

mismo, con lo cual se desea que cualquier persona pueda utilizar la interfaz de este agente (Figura 3.16).

FORMULARIO DE VALIDACION	
AddressBook Address:	
Introducir Nro de CI	723464
Nombre	Luis Angel
Apellido Paterno	Calle
Apellido Materno	Yujra
Edad	15
Verificacion de Duplicidad de voto	Habilitado para votar
Observaciones	*****Los datos se han validado*****
Validar	Enviar

Figura 3.15 Interfaz Grafica del Agente Servidor de la Mesa de sufragio.
Fuente: [Elaboración Propia]

Como se puede observar en la anterior grafica, el administrador de la mesa de sufragio solo tendrá que solicitar la cedula de identidad, con lo cual ingresará el numero de este documento y luego de presionar el botón “Validar” se mostrará los datos de la persona además de dejarle saber si la persona está habilitada o no para realizar la votación como lo muestra en la Figura 3.15 y la Figura 3.16.

FORMULARIO DE VALIDACION	
AddressBook	Address: atp://10.1.1.3/
Introducir Nro de CI	654854
Nombre	Juan Ernesto
Apellido Paterno	Siles
Apellido Materno	Garcia
Edad	28
Verificacion de Duplicidad de voto	Ya realizo su voto
Observaciones	*****Los datos se han validado***** *****Se ha enviado la papeleta de Votacion***** *****voto registrado***** *****Los datos se han validado***** *****Ya se realizo su voto no se puede enviar la papeleta*****
<div>Validar</div> <div>Enviar</div>	

Figura 3.16 Interfaz de Agente Servidor con su registro de acciones.
Fuente: [Elaboración Propia]

En la anterior figura también nos permite observar que se puede evitar que personas no habilitadas realicen su voto, esto brinda seguridad de que por cada persona solo se realice un voto.

3.6.5.3 Interfaz del Agente Servidor de Centralización

Se quiere que las personas que puedan acceder a este modulo puedan tranquilamente utilizarla sin necesidad de un conocimiento avanzado sobre programación o manejo de interfaces de sistemas complejos, es que así también se trató de desarrollar una interfaz amigable para que la centralización se realice rápidamente, también se encargo a esta interfaz de mostrar los resultados numerales de la votación después de haberse realizado la centralización de los datos (Figura 3.17).

FORMULARIO DE VALIDACION

INTRODUCIR LA DIRECCION DEL SERVIDOR A CENTRALIZAR DATOS

AddressBook Address:

Enviar Centralizar

RESULTADOS TOTALES DE LA VOTACION CENTRALIZADA

Nro de Votos del Partido 1	266
Nro de Votos del Partido 2	92
Nro de Votos del Partido 3	270
Nro de Votos del Partido 4	84
Nro de Votos Nulos	198

Observaciones

*****Los datos se han actualizado*****

Figura 3.16 Interfaz del Agente Centralizador. Fuente: [Elaboración Propia]

Esta imagen nos proporciona una interfaz que nos permite visualizar los datos tanto de la votación así como de los procesos que van ocurriendo en el transcurso del funcionamiento de este agente.

3.7 REQUERIMIENTOS DE HARDWARE Y SOFTWARE

Dado que el sistema está basado en una arquitectura de agentes Cliente/Servidor, los requerimientos de hardware y software se especifican de acuerdo a ellos.

SERVIDOR	
Hardware	Las características mínimas que se requieren son: Microprocesador Pentium IV de más de 2.0 Mhz., memoria RAM de 512MB o superior, disco duro de 60 GB como mínimo.
Sistema operativo	Windows 2000, XP, 2003, Vista y Seven , Linux , MacOS.

Programas Base	Plataforma J2SE con Java jdk 1.5 o superior
Librerías y servidor para aglets	Librerías de Aglets y servidor Tahití, Versión de Aglets 2.0.2 o superior
Otros Software	Gestor de Base de Datos Access

Tabla 3.1 Requerimientos del sistema para el funcionamiento de la plataforma. Fuente: [Elaboración Propia]

CLIENTE	
Hardware	Las características mínimas que se requieren son: Microprocesador Pentium III de más de 2.0 Mhz., memoria RAM de 256MB o superior, disco duro de 60 GB como mínimo, con tarjeta de red.
Sistema operativo	Se puede trabajar en Windows 2000 Profesional, 2003, XP, 7 Seven, Linux, MacOS.
Servidor Tahití	Servidor Tahití provista por Aglets 2.0.2 o superior

Tabla 3.2 Requerimientos del sistema para alojar al agente que hace de Cliente. Fuente: [Elaboración Propia]

4 DEMOSTRACIÓN DE LA HIPÓTESIS

4.2 ANTECEDENTES

Después del análisis de la problemática y el objetivo de la tesis, se tuvo que considerar los puntos principales en el desenvolvimiento de una verdadera elección, y se tomo como parámetro las elecciones presidenciales, referéndum de consulta sobre la Nueva Constitución Política del Estado de magistrados y elección de autoridades para el Órgano Judicial de la República de Bolivia, realizada en estos últimos años.

Estos procesos se lograron llevar a cabo según los reglamentos y leyes de la Antigua Constitución así como la nueva constitución Política del Estado a continuación se detallaran los artículos referidos a los plazos de entrega de resultados.

El siguiente es una artículo de la Ley N° 018, Ley del Órgano Electoral Plurinacional Promulgada por el presidente Evo Morales Ayma, Promulgada el 16 de Junio de 2010, escrita en su Capítulo II de Obligaciones y Atribuciones, en su Artículo 23 (Obligaciones) N° 9.

“Hacer conocer a la Asamblea Legislativa Plurinacional, en un plazo no mayor a los 30 días, los resultados oficiales de cada proceso electoral, referendo o revocatoria de mandato que haya sido organizado, dirigido, supervisado, administrado o ejecutado por el Órgano Electoral Plurinacional;”

El anterior artículo da por sentado que el resultado oficial se tendrá en no más de 30 días, lo cual resulta en demasiado tiempo, pero analizando el lanzamiento de los resultados finales después del último acto, el Órgano Electoral Plurinacional (OEP). Fue realizado el 16 de Octubre de 2011 y los resultados fueron entregados oficialmente el 7 de Noviembre del mismo año, esta diferencia nos da un tiempo de 22 días, en los que los funcionarios de Órgano Electoral Plurinacional lograron realizar el conteo al 100% de las mesas de sufragio de todo el país este es un promedio que aun es rápido considerando que en anteriores actos se utilizaron los 30 días disponibles según las leyes para la entrega de los resultados finales.

Otro factor que se pretende resolver es el excesivo costo de la realización de eventos electorales, para los cuales se necesita además de recursos económicos, material electoral que exige la producción innecesaria de papeletas de votación hechas de papel, ánforas hechas de cartón y otros accesorios que provocan la deforestación de más árboles, y por ende la contaminación del medio ambiente y la destrucción de buena parte de la naturaleza. Por ejemplo en las últimas elecciones realizadas el 16 de octubre se gastó alrededor de un poco más de 125 millones de bolivianos en el proceso completo incluyendo la elaboración de las papeletas y otros materiales.

4.3 COMPROBACIÓN

Según la información planteada en los antecedentes, se demostrará que si se puede optimizar el proceso de votación y obtención de resultados.

El proceso que se implantaría para la operación de la plataforma seguiría los siguientes pasos definidos por el proceso que se sigue en la actualidad , solo teniendo en cuenta las características que brinda un sistema de votación electrónica como es la nuestra mostrada en la Figura 4.1.

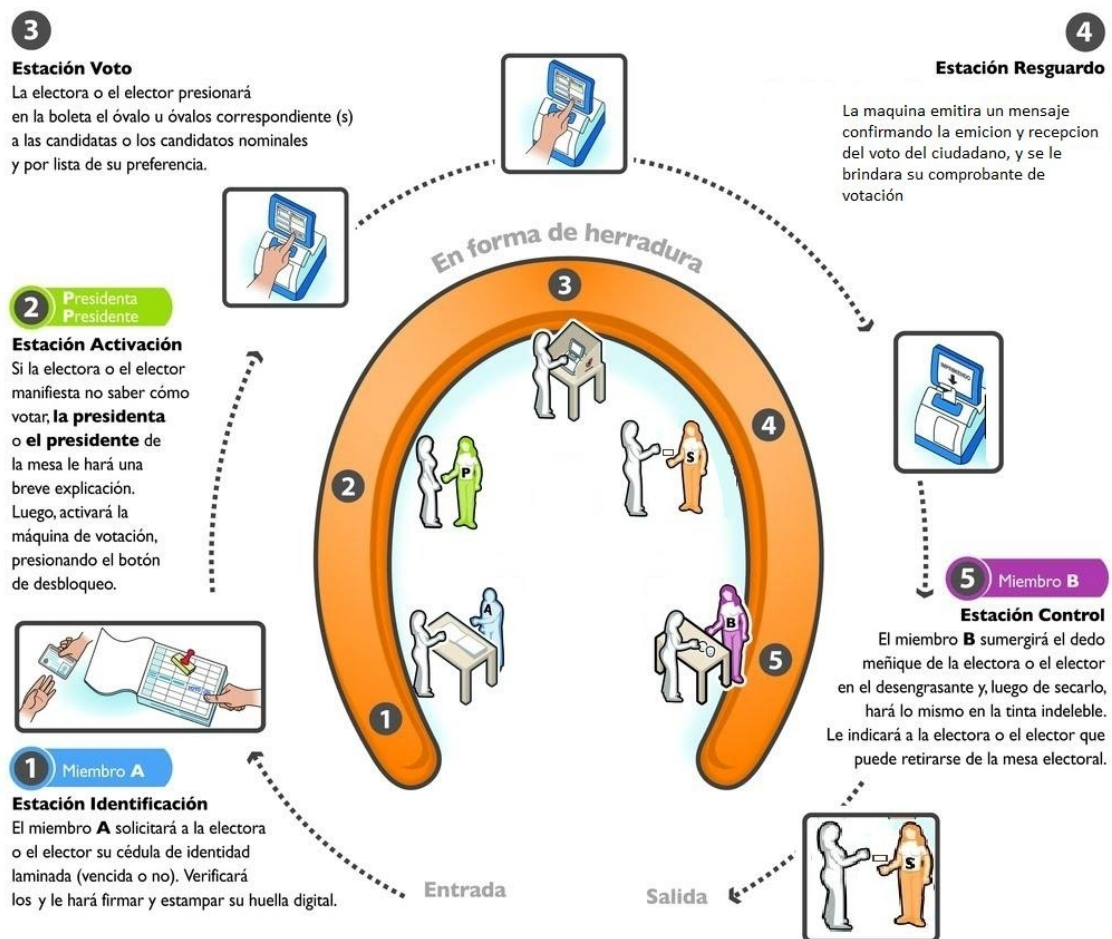


Figura 4.1 Modelo de pasos que se siguen en el proceso de votación.

[Actuar, 2009]

Con la implementación de la plataforma, el tiempo de verificación de los datos del votante es instantánea, no es necesario que se muestre las papeletas, así que el tiempo también se reduce en este aspecto, y al retirarse al votante se le entrega su carnet de sufragio y su cedula de identidad, de esta forma el proceso finaliza solo teniendo en cuenta el tiempo de decisión del votante.

Si consideramos el precio del material e insumos además de la contratación de los servicios, el precio se hace demasiado elevado, este precio puede elevarse al comprar las computadoras necesarias para realizar este proceso, pero se tiene que tener en consideración el beneficio que estas traerían a largo plazo, ya que pocas de ellas se desecharían, y se podrían utilizar en indefinidos procesos electorales, haciendo que el gasto por este insumo se realice una vez, dando una duración de varios años.

Podemos bien tomar para demostrar la reducción de costes análisis que se realizaron para el caso cercano de la argentina que fue realizado por el gobierno de ese país y otras instituciones con el fin de implantar un sistema de votación electrónica en el cual se demuestra lo siguiente:

Según la Asociación Actuar de la República de la Argentina presentada en su publicación "Gobierno Digital" es un hecho que la siguiente gráfica muestra tres momentos (A, B, C) que tienen los costos de la administración electoral con dos tipos de tecnologías aplicadas, el voto electrónico (e) y los procedimientos tradicionales (t) a lo largo de diversos procesos electorales (i, ii...n). Inicialmente (momento A), los costos del voto electrónico son mucho más elevados respecto al voto tradicional, pues toda instrumentación de nuevas tecnologías implica un costo superior respecto de la anterior por motivos diversos.

Una vez superados los escollos iniciales, la dinámica de la reutilización de los sistemas de votación electrónica significarían una reducción de los costos totales en los subsecuentes procesos electorales, llegando a un

momento (B) en el que los costos decrecen comparativamente respecto de los costos constantes de la tecnología tradicional. Éstos últimos son constantes en la medida en que prácticamente no cambian las modalidades de un proceso a otro y son costos elásticos a los cambios de precios de los insumos de un proceso a otro.

Las nuevas tecnologías aplicadas en los procesos electorales tienden a ser inelásticas respecto a las tradicionales porque, al ser reutilizables, el cambio de los insumos es menor de un proceso a otro (momento C). [Actuar, 2009]

Curvas de costos de la tecnología en los procesos electorales

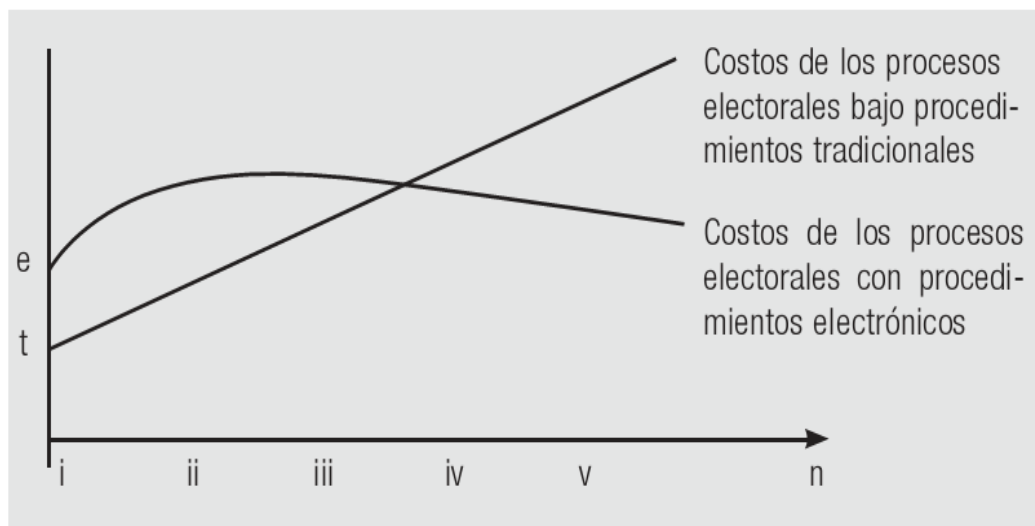


Figura 4.2 Grafica del comportamiento de costos de la Votación Electrónica
[Actuar, 2009]

Si comparamos los costes de la votación electrónica que se realiza en pueblos que implementaron este sistema en anteriores años y los comparamos conseguimos la siguiente tabla:

	Población Electora	Inversión (Dólares)	Dólares/Persona
Costo del Voto electrónico en Argentina(2008)	40519000	400000000	9,87
Costo del Voto electrónico en Brasil(2000)	115000000	600000000	5,22
Costo aprox. del voto electrónico en Paraguay(2013)	6550000	50000000	7,63
Costo presupuestado E- voto Costa Rica (2009)	4695000	25000000	5,32
Costo de Voto electrónico en Venezuela(2006)	28807000	200000000	6,94
Costo Promedio del E-voto en Latinoamérica			7,00

Tabla 4.1 Relación de Costos de la implementación del Voto Electrónico en Latinoamérica. Fuente: [Elaboración Propia]

Si consideramos los datos encontrados en la tabla 4.1 entonces podemos deducir el costo aproximado de la implementación de una plataforma de las mismas características y eso se puede hallar en la siguiente tabla:

Costo Promedio del E-voto en Latinoamérica por persona	7
Población electora en Bolivia(Según Padrón 2011)	5,138,583
Recursos E voto(\$us) [(Costo/persona)*Población]	35,960,278
Convirtiendo a Bs	251,721,948

Tabla 4.2 Costo Aproximado para implantación de Voto Electrónico. Fuente: [Elaboración Propia]

Ahora si contrastamos con los datos de la grafica de costos de las últimas elecciones Figura 4.3, podemos inferir que tiene un comportamiento similar al de la Figura 4.2, que está basada en la tabla 4.3.

Acto Electoral	Recursos Erogados (Bs)
7-May-2008 (Referéndum Revocatorio)	26000000
25-Ene-2009(Referéndum Nueva Constitución)	50000000
6-Dic-2009(Elecciones Generales Y Ref. Autonómico)	103000000
4-Oct-2010(Elecciones Departamentales y Municipales)	98343000
16-Oct-2011(Elecciones autoridades Órgano Judicial)	128546000

Tabla 4.3 Montos Erogados Para los diferentes Actos Electorales. Fuente: [Elaboración Propia]

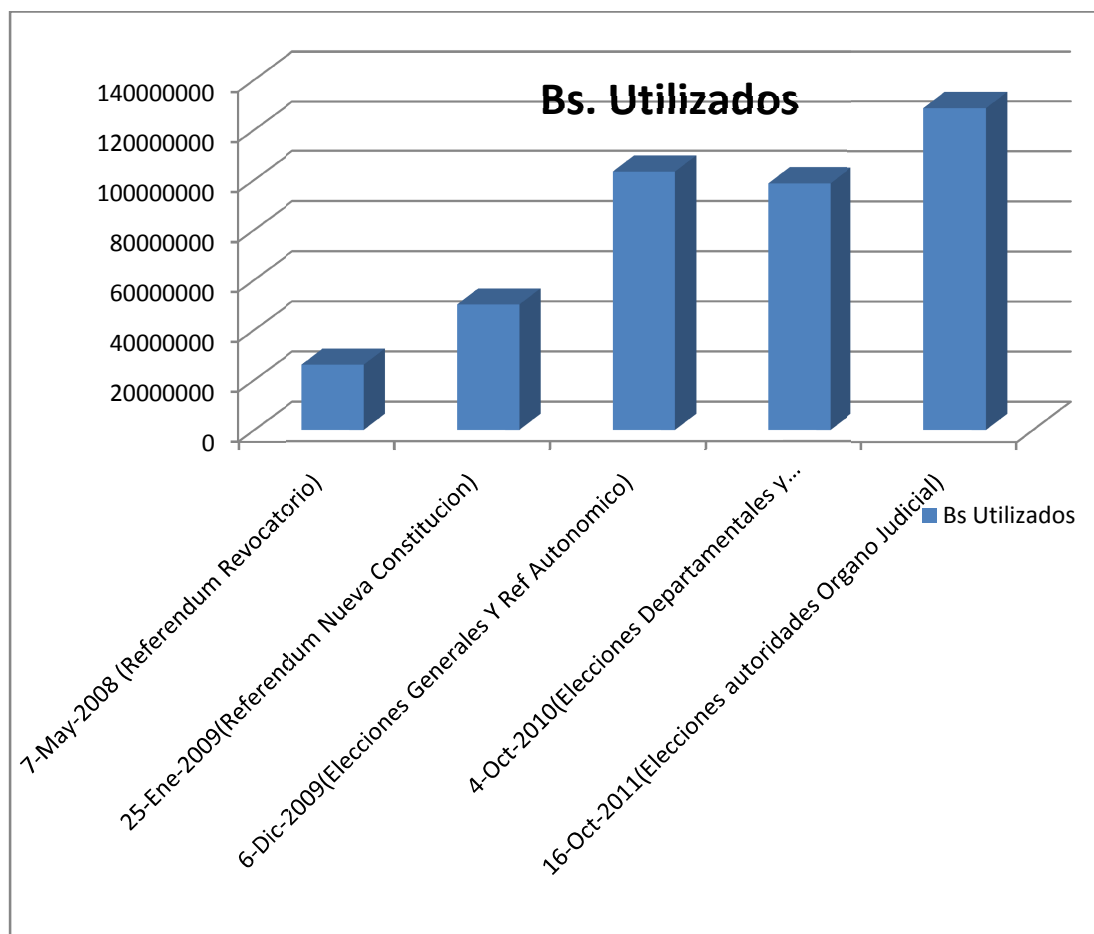


Figura 4.3 Montos erogados para elecciones en Bolivia. Fuente: [Elaboración Propia]

Para comprobar el número de actos electorales se necesitarán, para que los gastos se reduzcan de manera visible calcularemos está calculando el comportamiento de las elecciones tradicionales. Tomando como puntos (1,26) y (5,129)

Ecuación de la recta

$$103x - 103 = 4y - 104$$

Ahora se reemplaza el dato que obtuvimos 250 millones

$$103x - 103 = 4(250) - 104$$

$$x = 9$$

Por lo tanto en la novena elección ya se tendrá el punto donde los gastos empiezan a reducirse en comparación a la forma tradicional de elecciones, esto quiere decir que si tomamos en cuenta que ya se realizaron 5 elecciones faltarían 4 más si se implanta de inmediato para ver los beneficios económicos en el costo de la implantación de la plataforma.

La ventaja más importante que brinda la utilización de esta plataforma es el de lograr obtener los resultados del proceso en curso casi instantáneamente, pues los administradores de los servidores, pueden hacer un seguimiento a intervalos regulares o según sea necesario, de los resultados de la votación y de los porcentajes de estos, logrando así brindar la información lo más rápido posible, y ya que el tiempo es la mayor ventaja, y recurso de cualquier institución. Esta característica de la reducción de tiempo deriva en el siguiente análisis tabla 4.4:

Acto Electoral	Días Utilizados Para Obtención de Resultados Finales
7-May-2008 (Referéndum Revocatorio)	27
25-Ene-2009(Referéndum Nueva Constitución)	25
6-Dic-2009(Elecciones Generales Y Ref. Autonómico)	29
4-Oct-2010(Elecciones Departamentales y Municipales)	30
16-October-2011(Elecciones autoridades Órgano Judicial)	22
Una vez implantada la plataforma	0

Tabla 4.4 Tiempo que se utilizo para la obtención de resultados finales.
Fuente: [Elaboración Propia]

Con lo cual se puede demostrar por principio del absurdo, consideramos que:

$$x_1 = 27, x_2 = 25, x_3 = 29, x_4 = 30, x_5 = 22$$

$$Y, \quad y = 0$$

Por lo tanto afirmamos que $x_n \text{ tq } x < y \text{ con } n = 1,2,3,4,5$

Verificamos $\text{con } x_1 \quad 27 < 0 \quad \text{es Falso}$

$\text{con } x_2 \quad 25 < 0 \quad \text{es Falso}$

$\text{con } x_3 \quad 29 < 0 \quad \text{es Falso}$

$\text{con } x_4 \quad 30 < 0 \quad \text{es Falso}$

$\text{con } x_5 \quad 22 < 0 \quad \text{es Falso}$

Por lo tanto la afirmación es falsa y se demuestra el absurdo con lo cual se puede observar que la plataforma es más eficiente en la presentación de resultados finales a la población.

Es por todo lo explicado en este capítulo que se puede decir que la hipótesis se logró comprobar, en todos los aspectos, ya que se logró cumplir con el objetivo principal, así como los objetivos específicos. Con toda la información es fácil concluir que se pudo solucionar el problema que aquejaba, brindando una solución práctica, que ayudaría a la optimización de los procesos electorales que se realizan en el país. Este aporte, no solo brindaría una solución al país, sino a toda institución que necesita realizar una justa democrática, aportando a la sociedad una opción factible para solucionar problemas similares.

5 CONCLUSIONES Y RECOMENDACIONES

Luego de analizar los diferentes puntos desarrollados en el transcurso de desarrollo de la tesis, se pudo visualizar de una diferente perspectiva la problemática que aqueja el proceso electoral en el país, lo cual conlleva a un pensamiento retrograda, que hace pensar a la gente que estos procesos pueden seguir de igual forma, sin cumplir con la evolución necesaria para automatizar y optimizarlos.

Posteriormente se pudo llegar a la conclusión que es posible optimizar el proceso de votación y obtener resultados en un menor tiempo con la utilización de una plataforma basada en agentes, la cual brinda ventajas adicionales a la seguridad e innovación de tecnologías.

Conclusiones Derivadas del Trabajo.

- Por medio de la arquitectura planteada se pudo ver que los agentes móviles permiten replantear la forma en que se diseñan los procesos, permitiendo desacoplar procesos sin importar su ubicación.

- La aplicación de prueba mostró que los agentes móviles proveen un alto grado de autonomía, evitando la necesidad de un proceso “padre” que se encuentre monitoreando constantemente las actividades del agente para ver si éste cumple su misión.
- Mediante los agentes móviles se pueden comunicar sistemas heterogéneos en lugares distintos sin tener que estar creando interfaces de comunicación ni servidores dedicados a esto, como se puede observar al ser utilizados en diversos tipos de sistemas operativos.
- El proyecto desarrollado demostró que los agentes móviles son una inigualable oportunidad para reducir los costos en la comunicación, ya que pueden encapsular y filtrar los datos enviados de un lado a otro.
- Los agentes móviles se pueden adaptar fácilmente a su entorno, optimizando su ejecución sin tener que hacer grandes cambios en la arquitectura. Gracias a esto, el sistema puede crecer en el número de servicios y procesos sin sufrir cambios significativos.
- Las limitaciones de recursos de hardware no deben impedir que se sigan construyendo aplicativos para los dispositivos móviles, por que como se demostró en este proyecto, es necesario ver que con pocos recursos se pueden desarrollar muchas funcionalidades.

Se recomienda el uso de esta plataforma, tratando de primero instalarla en un institución de prueba para corroborar los defectos y subsanarlos, así como ver el desempeño, esta plataforma puede servir para el mejoramiento de los procesos electorales, los cuales son muy conflictivos en nuestra casa de estudios, remediando muchas de las falencias. Esta plataforma en el futuro puede servir de base para que en un futuro no muy distante, se pueda ser implementada por el órgano electoral de Bolivia, e incluso servir como un producto de exportación para su implementación en otros países con similares aspiraciones.

Otra recomendación nos da cuenta que se podría aprovechar las características de la nueva versión de aglets 2.5 para poder potenciar el sistema dando un aspecto más novedoso, incluyendo imágenes y la posibilidad de utilizar las nuevas librerías de java para el desarrollo de entornos amigables, con el propósito de facilitar las operaciones que la plataforma tiene que realizar.

BIBLIOGRAFÍA

- [Ahogado & Reinemer, 2003] Ahogado D., Reinemer A., “Programación Orientada a Agentes Metodologías de Desarrollo de Software”. Pontificia Universidad Javeriana, 2003.
- [Moulin y Chaib-draa, 1996] B. Moulin y B. Chaib-draa. An Overview of Distributed Artificial Intelligence. En G. O’Hare y N. Jennings, editores , Foundations of Distributed Artificial Intelligence, capítulo 1. John Wiley and Sons, 1996.
- [Finin et al., 1997] T. Finin, Y. Labrou, y J. Mayfield. KQML as an Agent Communication Language.
- [Fuggetta et al., 1998] Fuggetta, G. P. Picco, y G. Vigna. Understanding Code Mobility. IEEE Transactions on Software Engineering, 24(5):342–361, 1998. IEEE Press.
- [Booch y Rumbaugh, 2000] G. Booch, J. Rumbaugh, I. Jacobson. Addison Wesley Iberoamericana, 2000: “El Lenguaje Unificado de Modelado”.
- [Nwana, 1996] H. Nwana. Software Agents: An Overview. Knowledge Engineering Review, 11(3):205–244, 1996. Cambridge University Press.

- [Nwana y Azarmi, 1997] H. S. Nwana y N. Azarmi. Software agents and soft computing: towards enhancing machine intelligence: concepts and applications, tomo 1198 de Lecture Notes in Computer Science and Lecture Notes in Artificial Intelligence. Springer-Verlag, New York, NY, USA, 1997. ISBN 3-540-62560-7.
- [Parunak, 1996] H. Parunak. Go to the ant: Engineering principles from Natural MultiAgents Systems. En Annals of Operations Research. 1996.
- [Huhns y Singh, 1997] M. N. Huhns y M. P. Singh, editores. Readings in Agents. Morgan Kaufmann Publishers, 1997. ISBN 1-55860-495-2.
- [Lange y Oshima, 1998] Lange D. y Oshima M., "Mobile Agentes with Java: The Aglet API" World Wide Web, Septiembre de 1998
- [Bradshaw, 1997] J. M. Bradshaw. Software Agents. AAAI Press, Menlo Park, USA, 1997. ISBN 0-262-52234-9.
- [Minsky, 1985] M. Minsky. The Society of Mind. Simon and Schuster, New York, 1985.
- [Maes, 1995] P. Maes. Artificial Life Meets Entertainment: Lifelike Autonomous Agents. Communications of the ACM, 38(11):108–114, 1995. ACM Press. ISSN 0001-0782.
- [Rao y Georgeff, 1995] S. Rao y M. P. Georgeff. BDI Agents: from theory to practice. En V. Lesser, editor, Proceedings of the First International Conference on Multi-Agent Systems, páginas 312–319. MIT Press, San Francisco, CA, 1995.
- [Schmuller J.] Schmuller Joseph. "Aprendiendo UML en 24 Horas"
- [Sun Microsystems] Sun Microsystems Hispanoamérica, <http://ve.sun.com/service/sunps/jdc/jini.html>,

Noviembre 10 2002.

[The Foundation for
Intelligent Physical Agents,
1997]

The Foundation for Intelligent Physical Agents. The
FIPA'97 Specification. [http://drogo.
cselt.it/fipa/spec/fipa97/fipa97.htm](http://drogo.cselt.it/fipa/spec/fipa97/fipa97.htm), 1997.

[Demazeau y Muller, 1991]

Y. Demazeau y J. Müller. From Reactive to
Intentional Agents. En Y. Demazeau y J. Müller,
editores, Decentralized A.I. 2 — Proceedings of the
Second European Workshop on Modelling
Autonomous Agents in a Multi-Agent World
(MAAMAW-90), páginas 3–10. Elsevier Science
Publishers B.V., Amsterdam, The Netherlands,
1991.

[Beck K., 2000]

Beck, K.. "Extreme Programming Explained.
Embrace Change", Pearson Education, 1999.
Traducido al español como: "Una explicación de la
programación extrema. Aceptar el cambio", Addison
Wesley, 2000.

[Yágüez J, 2010]

Yágüez Javier. "Sistemas de desarrollo Cliente y
Servidor Orientados a Agentes Móviles".
Presentación de Diapositivas. Marzo de 2010

[Wake W., 2002]

Wake, W.C. "Extreme Programming Explored".
Addison-Wesley. 2002.

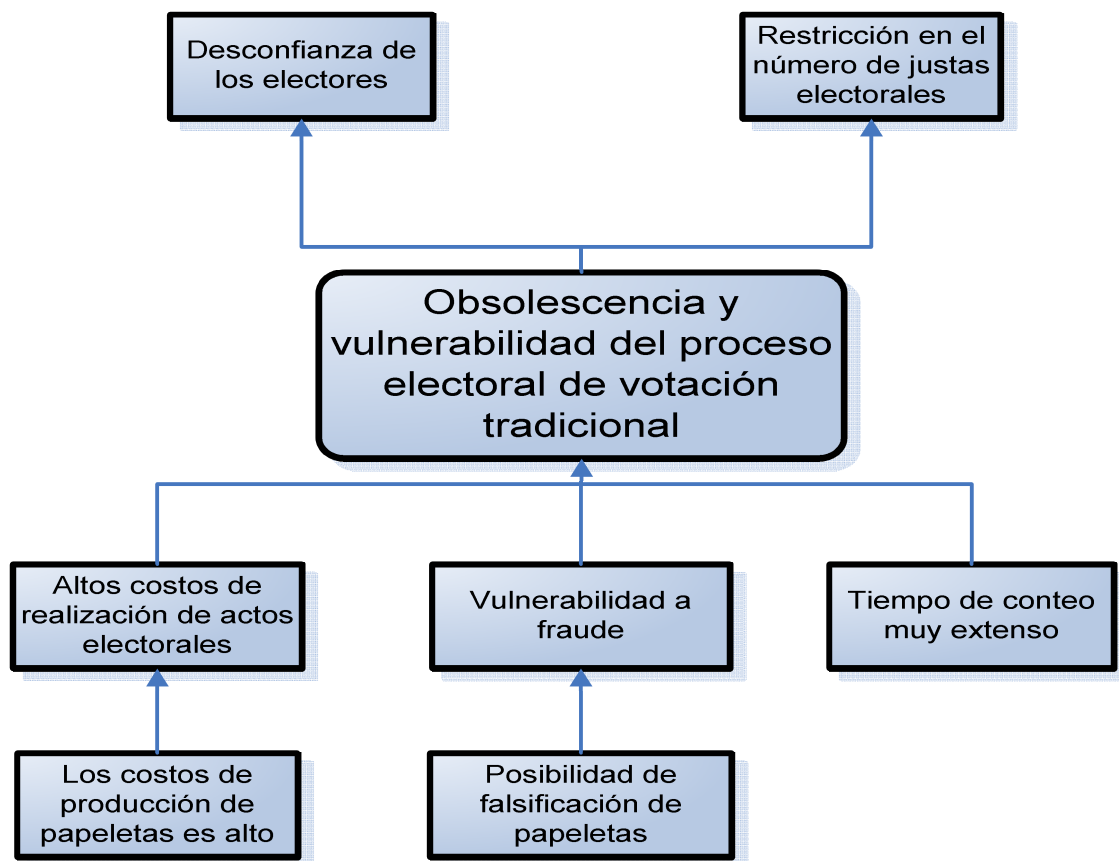
[Actuar, 2009]

Asociación Actuar República de Argentina
www.gobiernodigital.org.ar, "Gobierno Digital", 2009.

ANEXOS

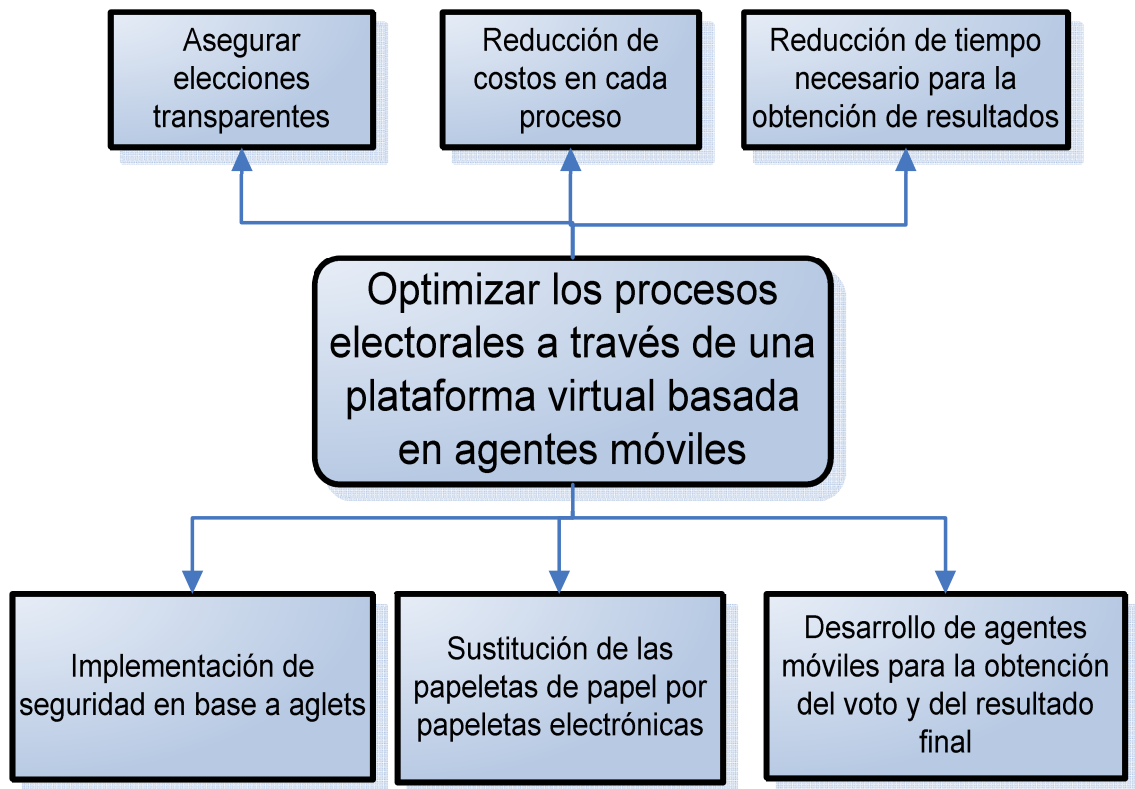
ANEXO 1

ÁRBOL DE PROBLEMAS



ANEXO 2

ÁRBOL DE OBJETIVOS



ANEXO 3

MATRIZ DEL MARCO LÓGICO

DESCRIPCIÓN	INDICADORES	VERIFICADORES	SUPUESTOS
Optimizar el proceso de votación, en diferentes aspectos relacionados a la confiabilidad, seguridad y reducción de tiempo del proceso de escrutinio.	Contribución a la población con la presentación de una herramienta útil que coadyuvara con los procesos electorales	Aprobación de los estándares propuestos por los estatutos o similares de las posibles instituciones que podrían utilizar el producto	Demostración de la fiabilidad y ventajas proporcionadas por el sistema
Reducir el costo y tiempo del proceso de elección y conteo de votos, además de asegurar el proceso de votación.	Al termino del sistema se lograra Una reducción en el tiempo de respuesta a la finalización de cualquier votación.	Aprobación de las diferentes pruebas realizadas al prototipo final	Aprobación del prototipo y de la documentación presentada

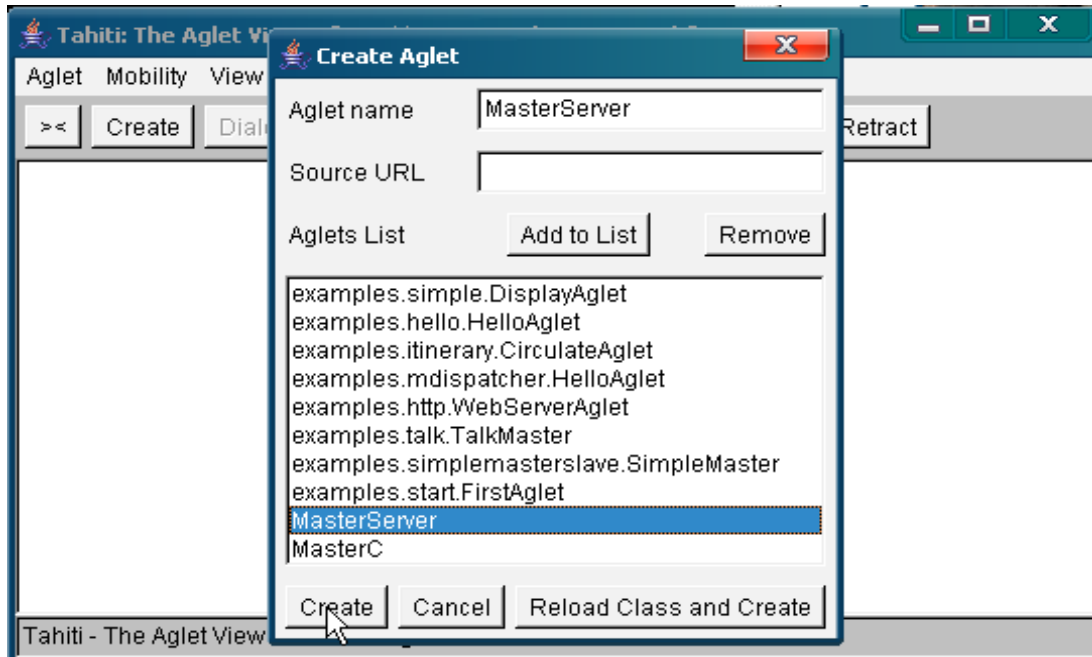
<p>Como resultado se tendrá un sistema que servirá como herramienta para la realización de procesos electorales, en diferentes instituciones donde sea requerido.</p>	<p>El prototipo será probado verificando su usabilidad en el contexto real realizándose pruebas de campo</p> <p>Así se realizara una comparación con los parámetros de los sistemas de votación actual.</p>	<p>Entrega de la documentación y versiones de prueba después de cada lapso de tiempo para la calificación del avance.</p>	<p>Implementación de los módulos relacionados al sistema.</p>
<p>Redacción y planificación de las actividades a realizarse</p> <p>Realizar la documentación del sistema</p> <p>Realizar los módulos principales del sistema</p> <p>Realizar los módulos de</p>	<p>La finalización y documentación que demuestre la realización de cada uno de los avances resultara en un gasto según cada etapa desarrollada y finalizada, para lo cual se proyecta un presupuesto total aproximado de 250 millones de Bs.</p>	<p>Demarcación y evaluación del seguimiento del cronograma establecido con el fin de finalizar en el tiempo establecido.</p>	<p>Aprobación y realización de cada una de las etapas de elaboración del proyecto</p>

secundarios			
Realización de las pruebas			
Realización de correcciones			
Realización de las pruebas finales			
Entrega del prototipo final			

ANEXO 4

INFORMACIÓN DEL SISTEMA

- Creación de un servidor : en este caso se crea un servidor de votación



- Agente Servidor Creado: se puede observar la ventana del servidor y de votación, así como el servidor Tahití donde se encuentran los datos del nuevo agente.

FORMULARIO DE VALIDACION

AddressBook Address:

Introducir Nro de CI

Nombre

Apellido Paterno

Apellido Materno

Edad

Verificacion de Duplicidad de voto

Observaciones

Validar Enviar

MasterServer : Thu Nov 10 14:50:57 COT 2011

- *Realización de la validación del elector: colocando el numero de su documento de identificación se puede verificar los datos del votante.*

FORMULARIO DE VALIDACION

AddressBook Address:

Introducir Nro de CI 723464

Nombre Luis Angel

Apellido Paterno Calle

Apellido Materno Yujra


Edad 15

Verificacion de Duplicidad de voto Habilitado para votar

Observaciones *****Los datos se han validado*****

Validar Enviar

- *Introducir la dirección de destino: en este paso se introduce la dirección del destino del host donde se realizara el envío de la papeleta virtual(agente esclavo). Este paso se realiza a un complemento de las librerías de aglets el cual es el AdressBook o libro de direcciones , donde se almacenan las direcciones o DNS's.*

 **FORMULARIO DE VALIDACION**

Close

Address:

AddressBook

Add to AddressBook

Delete

atp://10.1.1.3/

- *Envío de la Papeleta: en este caso la papeleta resulta siendo un Agente Esclavo que viaja al destino , para obtener la opción del votante, y una vez validados los datos del votante se presiona “Enviar”*

FORMULARIO DE VALIDACION	
AddressBook	Address: atp://10.1.1.3/
Introducir Nro de CI	723464
Nombre	Luis Angel
Apellido Paterno	Calle
Apellido Materno	Yujra
Edad	15
Verificacion de Duplicidad de voto	Habilitado para votar
Observaciones	*****Los datos se han validado***** *****Se ha enviado la papeleta de Votacion***** *****voto registrado*****
<div>Validar</div> <div>Enviar</div>	

- *Papeleta de sufragio*

Papeleta de Sufragio	
Partido 1	Partido 2
Partido1	Partido2
Partido 4	Voto Nulo
Partido4	Voto Nulo

- *El historial : en la parte inferior de la ventana del Agente Servidor de Votación se encuentra el historial de lo que sucede*

FORMULARIO DE VALIDACION

AddressBook Address: atp://10.1.1.3/

Introducir Nro de CI: 723464

Nombre: Luis Angel

Apellido Paterno: Calle

Apellido Materno: Yujra

Edad: 15

Verificacion de Duplicidad de voto: Habilitado para votar

Observaciones:

*****Los datos se han validado*****
 *****Se ha enviado la papeleta de Votacion*****
 *****voto registrado*****
 *****Los datos se han validado*****
 *****Ya se realizo su voto no se puede enviar la papeleta*****
 *****Los datos se han validado*****
 *****Se ha enviado la papeleta de Votacion*****

Validar Enviar

- *Centralización: El otro modulo nos presenta el proceso de centralización de datos, lo cual nos muestra a través de la siguiente ventana.*

FORMULARIO DE CENTRALIZACION

INTRODUCIR LA DIRECCION DEL SERVIDOR A CENTRALIZAR DATOS

AddressBook Address:

Enviar Centralizar TOTAL DE VOTOS

RESULTADOS TOTALES DE LA VOTACION CENTRALIZADA

Nro de Votos del Partido 1	1230	44	%
Nro de Votos del Partido 2	356	12	%
Nro de Votos del Partido 3	538	19	%
Nro de Votos del Partido 4	368	13	%
Nro de Votos Nulos	297	10	%

Observaciones:

*****Los datos se han actualizado*****